Trabajo Fin de Grado

Grado en Ingeniería de las Tecnologías de Telecomunicación

Integración de dispositivos domóticos Loxone y Z-wave sobre OpenHAB 3 y desarrollo de una interfaz gráfica 3D común para el control de una vivienda

Autor: Sara Callejón Sánchez

Tutor: Jesús Iván Maza Alcañiz

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Integración de dispositivos domóticos Loxone y Z-wave sobre OpenHAB 3 y desarrollo de una interfaz gráfica 3D común para el control de una vivienda

Autor:
Sara Callejón Sánchez

Tutor:
Jesús Iván Maza Alcañiz

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Proyecto Fin de Carrera: Integración de dispositivos domóticos Loxone y Z-wave sobre OpenHAB 3 y desarrollo de una interfaz gráfica 3D común para el control de una vivienda

Autor: Sara Callejón Sánchez

Tutor: Jesús Iván Maza Alcañiz

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

“El esfuerzo constante, no la fuerza o la inteligencia, es la clave para liberar nuestro potencial.”

Winston Churchill

Agradecimientos

A todas esas personas que estuvieron en mi vida durante mi etapa universitaria y me ayudaron durante la misma, compañeros de clase, amigos, familia, compañeros de trabajo, profesores y profesoras, todas aquellas personas que con un simple gesto o palabra me animaban para acabar este maratón intelectual.

En especial a Jorge, por todos esos días eternos en la ETSI intentando hacer la misma "magia" que nos mostraban en clase, gracias a tus explicaciones y a esos ratitos de desconexión que nos daban la vida entre tanto estudio.

Resumen

Actualmente el control automático del hogar y de los edificios está cada vez más demandado, aunque hoy día la mayoría de las soluciones son un poco simples, para las personas es suficiente con poder encender la luz sin tener que darle al interruptor. Para este tipo de utilidades existen en el mercado multitud de dispositivos, pero no siempre son compatibles unos con otros debido a sus características o al interés de las empresas de intentar que sus dispositivos solamente puedan comunicarse entre ellos para "atar" de cierta forma al usuario final.

Con este proyecto se demuestra la posibilidad de solventar este problema mediante el uso de OpenHAB 3. Para ello se creará una central domótica con OpenHAB 3, en la que se integrarán dispositivos Loxone y Z-wave, los cuales no están diseñador para comunicarse entre ellos.

También se ha diseñado e implementado una interfaz gráfica 3D mediante la que se podrá hacer uso de las funcionalidades ofrecidas por todos los dispositivos.

Abstract

Currently, the demand for the automatic control of houses and buildings is getting greater and greater. Although nowadays the majority of the solutions are a bit simple, people think it is enough with the possibility of turning the lights on without touching the switch. For these applications, there is a large range of devices in the market, but they are not always compatible with each other due to their characteristics or the businesses' interest of the companies behind to "tie" somehow the final user. This project shows the possibility to integrate home automation devices of different technologies and companies through the use of OpenHAB 3. An automated central has been designed and developed with OpenHAB 3 to integrate in the same framework Loxone and Z-wave devices which natively cannot communicate with each other.

Additionally, a 3D graphical user interface has been developed to monitor and control the different functionalities offered by all the devices.

Índice

Agradecimientos **9**

Resumen **11**

Abstract **13**

Índice **15**

Índice de Tablas **17**

Índice de Figuras **19**

1 INTRODUCCIÓN **22**

1.1 Motivación 22

1.2 Objetivos 22

1.3 Estructura de la memoria 22

2 OPENHAB 3 **24**

2.1 Introducción 24

2.2 Conceptos 25

2.2.1 Binding 25

2.2.2 Things 25

2.2.3 Channel 26

2.2.4 Items 26

2.2.5 Link 27

2.2.6 Sitemap 27

2.3 Dispositivos para openHAB 3 28

2.3.1 Raspberry Pi 28

2.4 Instalación de OpenHABian en Raspberry Pi 28

3 LOXONE **32**

3.1 Introducción 32

3.2 Dispositivos Loxone 33

3.2.1 Miniserver Gen. 1 33

3.2.2 Touch tree 33

3.2.3 Dimmer Tree RGBW y Tira Led 33

3.2.4 Sensor de Movimiento y Luminosidad tree 34

3.2.5 Contacto de Puerta/Ventana 34

3.2.6 Extension Air 34

3.2.7 Contacto de Puerta/Ventana Air 34

3.3 Instalación de los dispositivos Loxone en OpenHAB 3 35

4 Z-WAVE **44**

4.1 Introducción 44

4.2 Dispositivos Z-wave 45

4.2.1 Z-Stick Gen5 de Aeotec 45
4.2.2 Motion Sensor de Fibaro 46
4.2.3 Flood Sensor de Fibaro 46
4.3 Instalación de los dispositivos Z-wave en OpenHAB 3 46
5 INTERFAZ DE USUARIO 52
5.1 Plano de la vivienda en 3D 52
5.1.1 HABpanel 52
5.1.2 Sweet Home 3D 54
5.2 Creación de la interfaz en 3D 54
5.2.1 Instalación del plano de Sweet Home 3D en HABpanel 54
5.2.2 Asociar un Item a un dispositivo virtual del plano 57
5.3 IU Basic 59
5.4 Creación de la interfaz en IU Basic 59
6 CONCLUSIONES Y LÍNEAS DE FUTURO 62
Referencias 63
Anexo 64

ÍNDICE DE TABLAS

Tabla 1. Estados de un Thing. 25
Tabla 2. Tipos de Item 26
Tabla 3. Tipos de elementos en Sitemap 27
Tabla 4. Tipos de Widgets 53

ÍNDICE DE FIGURAS

Ilustración 1. Logotipo de la iniciativa OpenHAB 24
Ilustración 2. Comunicación openHAB con los dispositivos 24
Ilustración 3. Relación entre conceptos esenciales de openHAB 28
Ilustración 4. Raspberry Pi 4B 28
Ilustración 5. Proceso de instalación OpenHABian 29
Ilustración 6. Pantalla inicial OpenHABian 29
Ilustración 7. Menú de configuración OpenHABian 30
Ilustración 8. Pantalla para crear el primer administrador 30
Ilustración 9. Pantalla de inicio de openHAB 3 31
Ilustración 10. Miniserver Gen.1 33
Ilustración 11. Touch tree 33
Ilustración 12. Dimmer Tree RGBW 33
Ilustración 13. Tira Led 33
Ilustración 14. Sensor de movimiento y luminosidad tree 34
Ilustración 15. Contacto de Puerta/Ventana 34
Ilustración 16. Extension Air 34
Ilustración 17. Contacto Puerta/Ventana Air 34
Ilustración 18. Pantalla de Configuración 35
Ilustración 19. Pantalla de Binding instalados 35
Ilustración 20. Instalación Binding de Loxone 36
Ilustración 21. Pantalla de Binding con Loxone instalado 36
Ilustración 22. Pantalla de Things instalados 37
Ilustración 23. Elección del Binding para instalar un Thing 37
Ilustración 24. Pantalla para añadir un Thing de Loxone 38
Ilustración 25. Configuración de Loxone Miniserver 1 38
Ilustración 26. Configuración de Loxone Miniserver 2 39
Ilustración 27. Pantalla de Things instalados con el Miniserver de Loxone 39
Ilustración 28. Activación del Thing del Miniserver 40
Ilustración 29. Estado Online del Thing del Miniserver 40
Ilustración 30. Channels del Miniserver 41
Ilustración 31. Creación de un Item 41
Ilustración 32. Channel con un Item asociado 42
Ilustración 33. Item Lámpara de Color 42
Ilustración 34. Items Loxone 43

Ilustración 35. Z-Stick Gen5 Aeotec 45
Ilustración 36. Motion Sensor Fibaro 46
Ilustración 37. Flood Sensor Fibaro 46
Ilustración 38. Instalación del Binding de Z-wave 47
Ilustración 39 Creación del Thing del Controlador de Z-wave 47
Ilustración 40. Selección del Serial Port del Stick 48
Ilustración 41. Thing del controlador de Z-wave instalado 48
Ilustración 42. Búsqueda de dispositivos Z-wave 49
Ilustración 43. Añadiendo el Motion Sensor 49
Ilustración 44. Configuración de parámetros del Motion Sensor 50
Ilustración 45. Channels del Motion Sensor 50
Ilustración 46. Creación del Item de Temperatura 51
Ilustración 47. Items Z-wave 51
Ilustración 48. Terminología de entidades de HABpanel 52
Ilustración 49. Activación de Samba 54
Ilustración 50. Contenido de la carpeta "sweethome3d" 55
Ilustración 51. Colocar plano en 3D en la carpera correspondiente 55
Ilustración 52. Pantalla inicial de HABpanel 56
Ilustración 53. Dashboard para la interfaz 3D 56
Ilustración 54. Agregación del widget Modelo 56
Ilustración 55. Importación de "3d-view.tpl.html" 57
Ilustración 56. Editar widget 57
Ilustración 57. Modificación de la ruta de los archivos 57
Ilustración 58. Ejemplo de un Item asociado a una lámpara 58
Ilustración 59. Visualización del control de la luminaria 58
Ilustración 60. Selección del tipo de widget 59
Ilustración 61. Widget para las persianas 60
Ilustración 62. Widget para el sensor de movimiento 60
Ilustración 63. Interfaz UI Basic 61

1 INTRODUCCIÓN

1.1 Motivación

Desde el principio de los tiempos el ser humano siempre ha utilizado la ciencia y la tecnología para usarla en su propio beneficio, ya sea para crear robots para la automatización de las fábricas hasta la creación de la lavadora para no perder su tiempo en algunas tareas tediosas.

Actualmente el ser humano va más allá y busca la automatización inteligente de su hogar o de cualquier edificio para ahorrar dinero y no perder su tiempo en encender una luz o bajar las persianas.

Debido al nuevo auge de querer automatizar lo que nos rodea, nos encontramos con multitud de sistemas domóticos diferentes, los cuales no son compatibles unos con otros, ya que cada cual tiene su propio protocolo de comunicación por lo que es necesario un sistema con el que poder automatizar una vivienda independientemente de los dispositivos usados y del sistema de comunicación que estos necesiten.

Esta necesidad puede ser solventada por OpenHAB, un software que nos permite el uso de dispositivos diferentes, independientemente de su protocolo de comunicación. En este proyecto se demuestra como mediante OpenHAB se puede crear un sistema domótico en el que se controlan dispositivos Loxone y Z-wave en perfecta armonía y desde una misma interfaz.

1.2 Objetivos

Los cuatro objetivos principales de este trabajo son los siguientes:

- Integrar el sistema Loxone en OpenHAB 3 y poder controlar sus dispositivos mediante la interfaz de OpenHAB 3 sin dificultad.
- Integrar dispositivos Z-wave en OpenHAB 3 y poder controlarlos mediante la interfaz de OpenHAB 3 correctamente.
- Demostrar la posibilidad de tener dispositivos Loxone y Z-wave mediante OpenHAB 3 dentro de un marco común de reglas de automatización.
- Manejar todos estos dispositivos mediante una interfaz gráfica 3D que simule la vivienda en la que estarían instalados.

1.3 Estructura de la memoria

En este apartado se resume la estructura de la presente memoria y se hace una breve descripción de cada uno de los capítulos.

La estructura de este proyecto está basada en seis capítulos:

- **Capítulo 1. Introducción.**

  Se comenta la motivación para realizar este proyecto, los objetivos que se quieren cumplir y la estructura de este**.**

- **Capítulo 2. OpenHAB 3.**

  Se explica qué es OpenHAB y sus características. Se describen los dispositivos usados para su instalación y por ultimo los pasos seguidos para instalar el sistema operativo OpenHABian.

- **Capítulo 3. Loxone.**

  Aquí se introduce el sistema Loxone y se comenta su funcionamiento. Se describen los dispositivos Loxone usados en este proyecto y se indican los pasos para poder hacer uso de ellos desde OpenHAB 3.

- **Capítulo 4. Z-wave.**

  En este capítulo se presenta el sistema Z-wave y se explica su funcionamiento. Se describen los dispositivos Z-wave usados en este proyecto y se indican los pasos para poder hacer uso de ellos desde OpenHAB 3.

- **Capítulo 5. Interfaz de usuario.**

  Aquí se explican las diferentes interfaces de usuario utilizadas para mostrar el correcto funcionamiento de los dispositivos mencionados anteriormente una vez están asociados a OpenHAB 3. Se indican los pasos seguidos para su creación.

- **Capítulo 6. Conclusiones y líneas de desarrollo futuro.**

  Finalmente se explican las concusiones obtenidas durante la realización del proyecto y las futuras líneas de investigación que serían interesante abordar.

2 OPENHAB 3

En este capítulo se describe qué es openHAB 3 y se explican conceptos fundamentales del mismo. Posteriormente se describen los dispositivos usados, así como los pasos seguidos para la instalación de OpenHABian.

2.1 Introducción

OpenHAB [1] es un software, de código abierto, que nos permite integrar diferentes sistemas de automatización de viviendas, dispositivos y tecnologías. Su logotipo se muestra en la Ilustración 1.

Ilustración 1. Logotipo de la iniciativa OpenHAB

Es independiente del proveedor, del hardware y del protocolo de comunicación que integre, siendo este uno de sus puntos fuertes, ya que nos permite integrar dispositivos de diferente fabricante bajo un mismo software.

Funciona en múltiples plataformas como Linux, Windows y OS X permitiendo su instalación en dispositivos de bajo coste como la Raspberry Pi.

Está desarrollado en Java y usa Apache Karaf junto con Eclipse Equinox para crear un entorno de ejecución OSGi convirtiéndolo en un software altamente modular que se extiende mediante add-ons, los cuales, le dan una amplia gama de competencias, desde la capacidad de crear interfaces de usuario, hasta la de interactuar con un gran número de dispositivos.

La comunicación de openHAB con los dispositivos, representada en la Ilustración 2, se realiza de forma asíncrona a través de un bus de eventos o puede haber comunicaciones con estado.

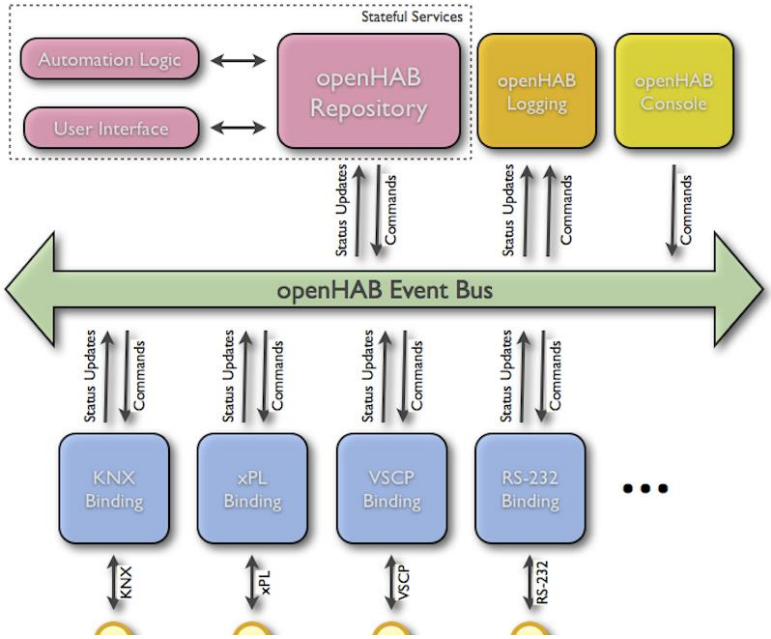

Ilustración 2. Comunicación openHAB con los dispositivos

Una de las modificaciones de openHAB 3 con respecto a las versiones anteriores es la posibilidad de hacer la configuración mediante una interfaz de usuario y no solo de forma textual, por lo que usaremos este nuevo modo para el desarrollo de este proyecto.

2.2 Conceptos

A continuación, se explican diferentes conceptos esenciales de openHAB para comprender su funcionamiento.

2.2.1 Bindings

Son enlaces que permiten el envío y recepción de comandos y actualizaciones de estado de los Things a través del bus de eventos de openHAB. Se puede considerar como un adaptador de software que permite que los Things estén disponibles para el sistema.

2.2.2 Things

Son entidades que pueden ser agregadas a nuestro sistema, ya sean dispositivos físicos, servicios web o cualquier fuente de información o funcionalidad que se pueda integrar. Pueden proporcionar más de una función dependiendo del Thing. Cada Thing tiene un estado y en la Tabla 1 se describen cada uno de sus posibles estados.

Estado	Descripción
Uninitialized	Es el estado inicial de un Thing al agregarse o iniciar el sistema. También es el estado cuando falla el proceso de iniciación o el Binding no está disponible por lo que los comandos enviados a sus Channels no se procesarán.
Initializing	Es el estado cuando el Thing se está iniciando. Los comandos siguen sin procesarse.
Unknown	El controlador está inicializado, pero debido al dispositivo o servicio representado, todavía no se sabe si el Thing está Online u Offline.
Online	El dispositivo o servicio representado por un Thing funciona correctamente y puede procesar comandos.
Offline	El dispositivo o servicio representado por un Thing no funciona correctamente y puede que no procese los comandos.
Removing	El dispositivo o servicio representado por un Thing debe eliminarse, pero el Binding aún no ha confirmado la eliminación.
Removed	Indica que el dispositivo o servicio representado por un Thing se eliminó del sistema externo después de que el sistema iniciara la eliminación.

Tabla 1. Estados de un Thing

2.2.3 Channel

Representan las funciones que proporcionan los Things, entendiendo que el Thing representa la entidad y el Channel es una función concreta de ese Thing.

2.2.4 Items

Representan capacidades que las aplicaciones pueden usar. Tienen un estado y pueden recibir comandos, los cuales pueden hacer cambiar el estado del Item.

En la Tabla 2 se definen los tipos de Item que existen.

Tipo	Descripción	Tipos de datos aceptados	Tipos de comandos aceptados
Call	Para elementos relacionados con las funcionalidades de telefonía.	Call	
Color	Para valores de color.	On/Off y porcentaje	On/Off, incremento/decremento y porcentaje
Contact	Para elementos que tengan como estado “abierto” o “cerrado”.	Open/Closed	
DateTime	Almacena un dato de tiempo.	DateTime	DateTime
Dimmer	Acepta porcentajes para establecer el valor.	On/Off, porcentajes	On/Off, incremento/decremento, porcentajes
Group	Colección de ítems.	Acepta el tipo de datos de los ítems a los que hace referencia	Acepta el tipo de comandos de los ítems a los que hace referencia
Location	Puede ser utilizado para almacenar información relacionada con GPS, direcciones, etc.	Coordenadas	Coordenadas
Number	Puede ser usado para cualquier tipo de sensor numérico como un contador.	Número	Número
Rollershutter	Permite el control de las persianas moviendo hacia arriba, abajo y parando o estableciendo un porcentaje	Up/Down, porcentaje	Up/Down, StopMove, porcentaje
String	Puede ser usado por cualquier tipo de cadena de texto o representación textual de una fecha	String, fecha	String
Switch	Representa a cualquier elemento que puede estar en estado ON o en estado OFF	On/Off	On/Off

Tabla 2. Tipos de Item

2.2.5 Link

Es una asociación entre un Channel y un Item. Los Channels pasan al estado "habilitado" cuando se les vincula a un Item. Los Item pueden estar vinculados a varios Channels y viceversa.

2.2.6 Sitemap

Es donde se enumeran los ítems que queremos mostrar en la interfaz de usuario y permite ver el estado de estos, así como su posible manipulación.

En la

Tabla 3 se describen los diferentes elementos que se puede utilizar en un Sitemap.

Elemento	Descripción
Colorpicker	Permite al usuario elegir el color
Chart	Agrega un objeto grafico para mostrar los datos registrados
Frame	Área con otros elementos de Sitemap o marcos anidados adicionales
Group	Agrupa los elementos definidos en un grupo
Image	Muestra una imagen
Selection	Da acceso a una nueva página donde el usuario puede elegir entre los valores definidos
Setpoint	Muestra un valor y permite al usuario modificarlo. Se pueden especificar valores máximos y mínimos
Slider	Control deslizante
Switch	Interruptor
Text	Texto
Video	Muestra un video
Webview	Muestra una página web

Tabla 3. Tipos de elementos en Sitemap

En la Ilustración 3 se representa la relación entre los conceptos anteriormente explicados:

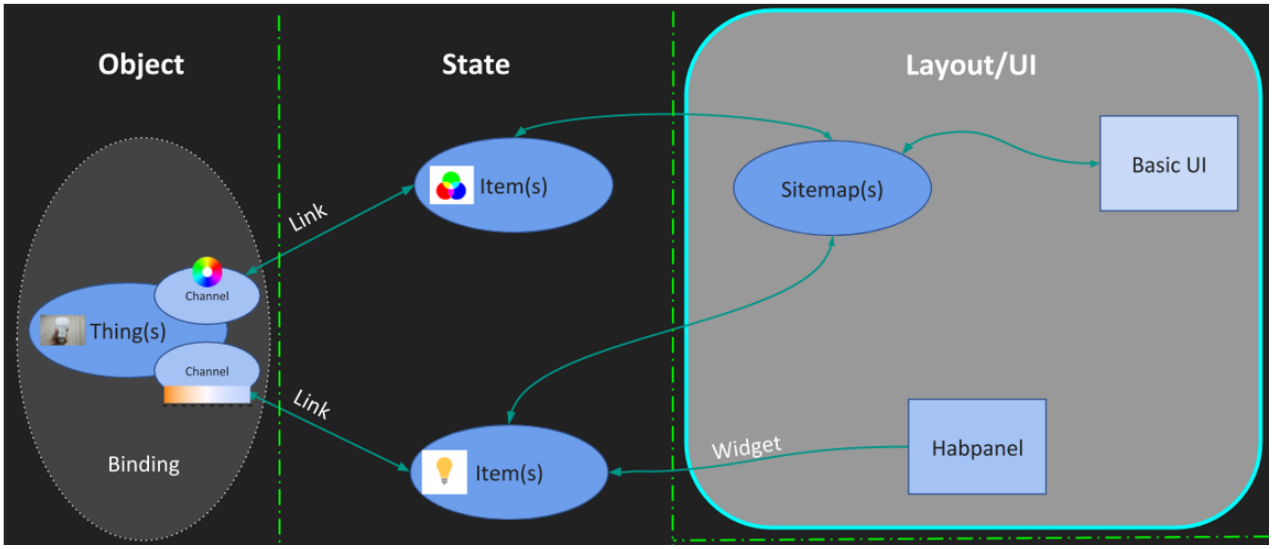

Ilustración 3. Relación entre conceptos esenciales de OpenHAB

2.3 Dispositivos para openHAB 3

2.3.1 Raspberry Pi

Se usó una Raspberry Pi 4B, representada en la Ilustración 4, para instalar OpenHABian y poder hacer uso de OpenHAB 3.

La Raspberry Pi es un ordenador de bajo coste y tamaño reducido. La Raspberry Pi 4B está compuesta por una CPU de cuatro núcleos a 1,5 GHz con brazo Cortex-A72, una GPU VideoCore VI, 4 GB de RAM, conectividad Wi-Fi, Bluetooth 5.0 y Gigabit Ethernet, 2 puertos micro-HDMI, 2 puertos USB 3.0 y otros 2 USB 2.0, una expansión de cabezal GPIO de 40 pines y se alimenta vía USB-C a 5V/3A.

Ilustración 4. Raspberry Pi 4B

2.4 Instalación de OpenHABian en Raspberry Pi

Los pasos seguidos a la hora de instalar OpenHABian [7] en la Raspberry Pi han sido los siguientes:

1. Descarga del archivo de la imagen de OpenHABian desde la página de openHAB.
2. Descarga e instalación de Raspberry Pi Imager, selección de la imagen de OpenHABian en el apartado de “Operating System” y selección de la tarjeta microSD, que usaremos posteriormente en la RaspberryPi, en “Storage” y pulsar en “WRITE” para la grabación de la imagen de OpenHABian en la microSD.

3. Insertar la microSD en la Raspberry Pi, conectarla al router mediante un cable Ethernet y encenderla.

4. Esperar hasta que OpenHABian se configure, lo cual tarda entre 15 y 45 minutos en completar todos los pasos. Es posible seguir el proceso mediante la conexión de la Raspberry Pi a un monitor o televisor o mediante la interfaz web en https://"ipopenHABian"/ o http://openhabian/, siempre que ese nombre esté disponible y no se haya modificado en el archivo openhabian.conf antes de iniciar la configuración de OpenHABian. El progreso de la instalación se muestra en la ilustración 5.

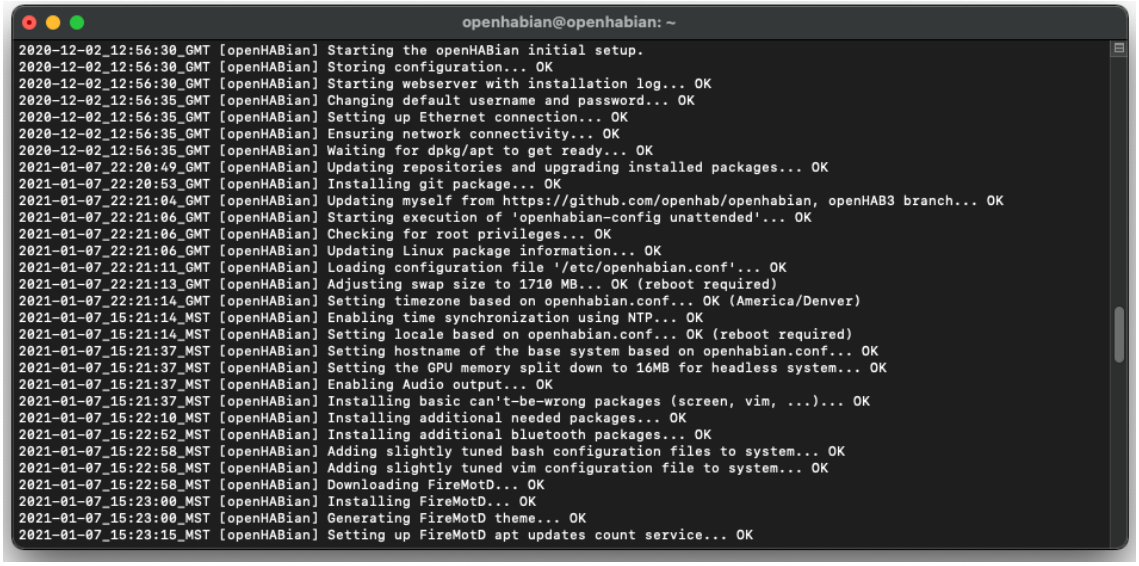

Ilustración 5. Proceso de instalación OpenHABian

5. Una vez finalizada la configuración, se puede acceder a OpenHABian mediante una sesión SSH. Usando PuTTY con la IP asignada a OpenHAB o el nombre de host "openhabian" y usando openhabian tanto para el nombre de usuario como para la contraseña, se accede al sistema y muestra una pantalla similar a la de la ilustración 6.

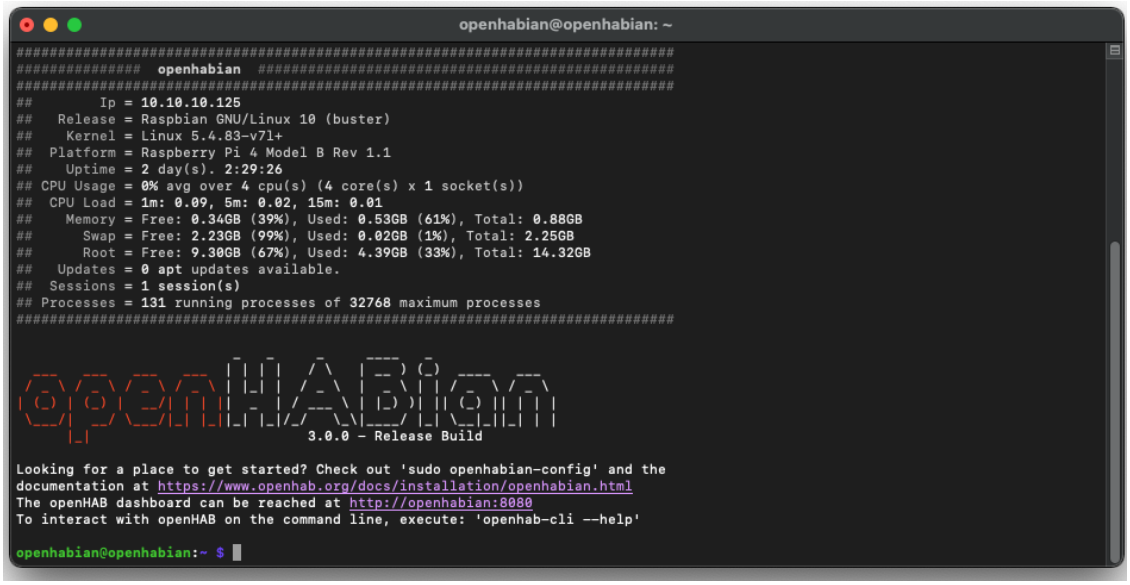

Ilustración 6. Pantalla inicial OpenHABian

Una vez llegado a este punto se puede ejecutar el comando “sudo openhabian-config” y aparece la ventana de la ilustración 7.

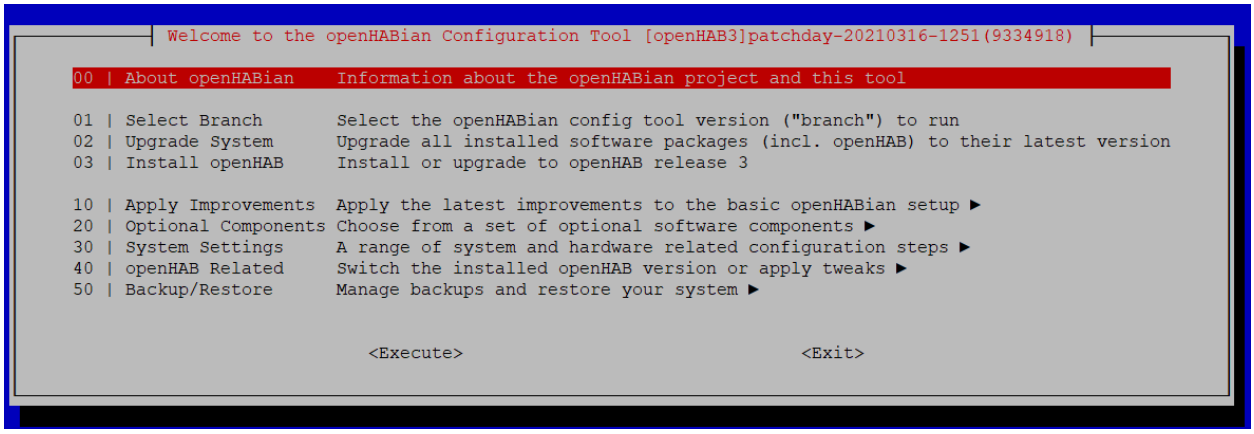

Ilustración 7. Menú de configuración OpenHABian

Desde esta herramienta se pueden configurar los componentes opcionales como Samba, crear o restaurar backup del sistema, actualizar el sistema o conectar la Raspberry Pi a nuestra red Wifi y no tener la obligación de tener la Raspberry Pi conectada al router por cable.

Una vez comprobado que el sistema esté actualizado, mediante un navegador se puede entrar en http://openhabian:8080 y aparece la pantalla de la ilustración 8, en la que se solicita el nombre para la cuenta de administrador y la contraseña para esa cuenta.

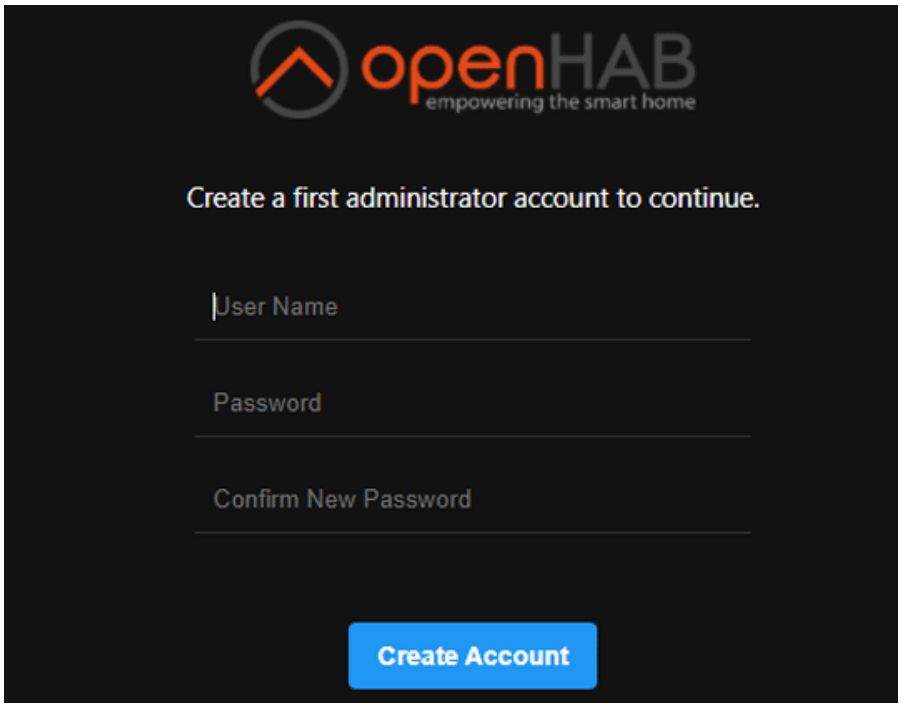

Ilustración 8. Pantalla para crear el primer administrador

En la siguiente pantalla pide configurar el idioma, región, zona horaria y ubicación y después de esta pantalla nos pregunta si queremos instalar Add-ons o instalarlos más tarde.

Finalmente llega a la página principal, como aparece en la ilustración 9, y a partir de aquí se empieza a configurar openHAB 3 para la comunicación con los dispositivos Loxone y Z-wave, por lo que la instalación de OpenHABian en la Raspberry Pi se da por finalizada correctamente.

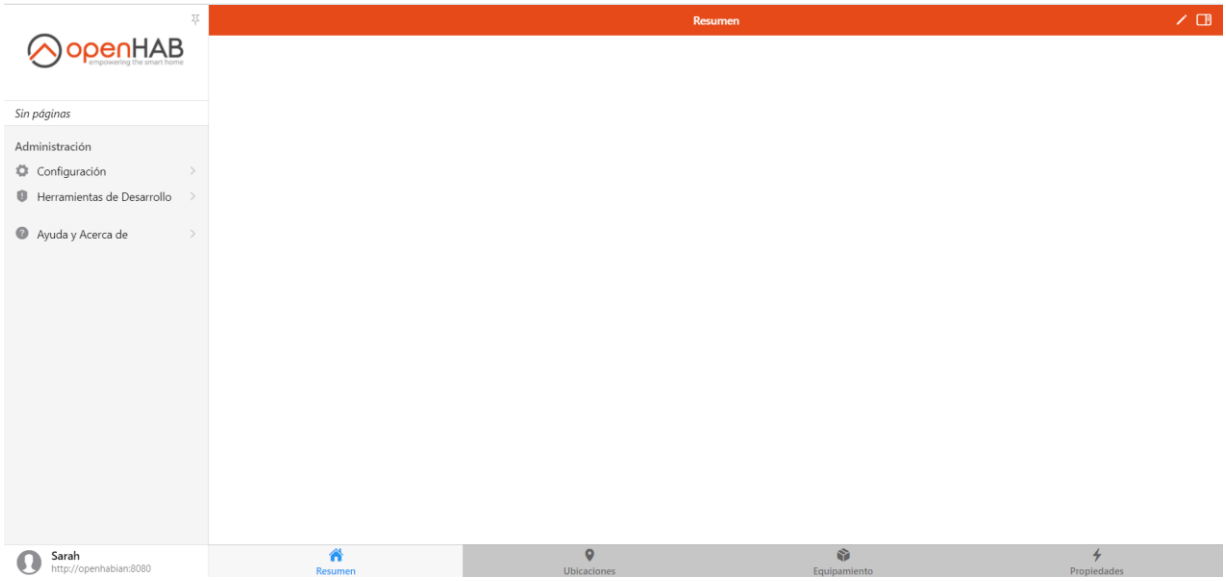

Ilustración 9. Pantalla de inicio de openHAB 3

3 LOXONE

En este capítulo se describe el sistema Loxone y se explican conceptos fundamentales del mismo. Posteriormente se describen los dispositivos usados para controlarlos mediante openHAB 3, así como los pasos seguidos para la vinculación de dichos dispositivos con openHAB 3.

3.1 Introducción

Loxone [2] es una empresa creada en 2009 en Austria dedicada a la automatización de edificios. Su sistema domótico, denominado Smart Home, tiene su propio protocolo de comunicación, cuenta con servidor web integrado y lenguaje de programación abierto.

Su sistema Smart Home es un sistema centralizado, donde los datos de los dispositivos son enviados al Miniserver y son procesados en él, así como todas las ordenes son enviadas desde el Miniserver hacia los diferentes dispositivos que integran el sistema.

Dispone de muchos dispositivos propios, como luminarias o pulsadores, y también es compatible con dispositivos de otros fabricantes, ya que es posible su comunicación con otros protocolos como por ejemplo KNX o Modbus.

Loxone dispone de dos tecnologías para la instalación de sus dispositivos:

Tecnología Tree: Sus dispositivos son alimentados y conectados al Miniserver bajo un mismo cable formado por 6 hilos, 2 hilos para la comunicación con el Miniserver, 2 hilos para la alimentación de dispositivos de muy bajo consumo, como por ejemplo sus pulsadores, y otros 2 hilos para la alimentación de dispositivos de mayor consumo, como son las luminarias. Esta conexión puede ser realizada en topología estrella, línea, árbol o bus.

La información por el bus de datos esta encriptada y la comunicación es bidireccional. Mediante este bus el Miniserver realiza la búsqueda de sus dispositivos para su integración hasta un máximo de 50 dispositivos por rama.

Al hacer la instalación con esta tecnología de forma total en una vivienda se reduce la instalación de cableado un 80 % respecto a una instalación eléctrica estándar.

Tecnología Air: La comunicación de los dispositivos con el Miniserver se realiza de forma inalámbrica en la frecuencia 868 MHz. Cada dispositivo que recibe tensión de forma permanente tiene una función repetidora por lo que se crea una red Mesh. La información se envía encriptada evitando posibles conflictos entre Miniservers. Tiene un máximo de 128 dispositivos por extensión Air.

Esta tecnología está pensada fundamentalmente para hacer rehabilitaciones en edificios o viviendas, ya que se usa la red eléctrica cableada existente para alimentar los dispositivos.

3.2 Dispositivos Loxone

3.2.1 Miniserver Gen. 1

Es el encargado de comunicar los sistemas y dispositivos que componen el sistema Loxone entre sí.

Se representa en la ilustración 10.

Tiene un procesador de 400 MHz y 64MB de memoria RAM, de la que usa unos 10 MB para el sistema operativo LoxOS con servidor Web.

Está formado por:

- 8 entradas digitales 8VDC – 24VDC
- 4 entradas analógicas 0-10V
- 4 salidas analógicas 0-10V (máx. 20mA)
- 8 salidas digitales 250V/5A
- Ranura para tarjetas SD
- Puerto LAN
- Puerto KNX/EIB
- Loxone Link
- Entrada de alimentación a 24VDC

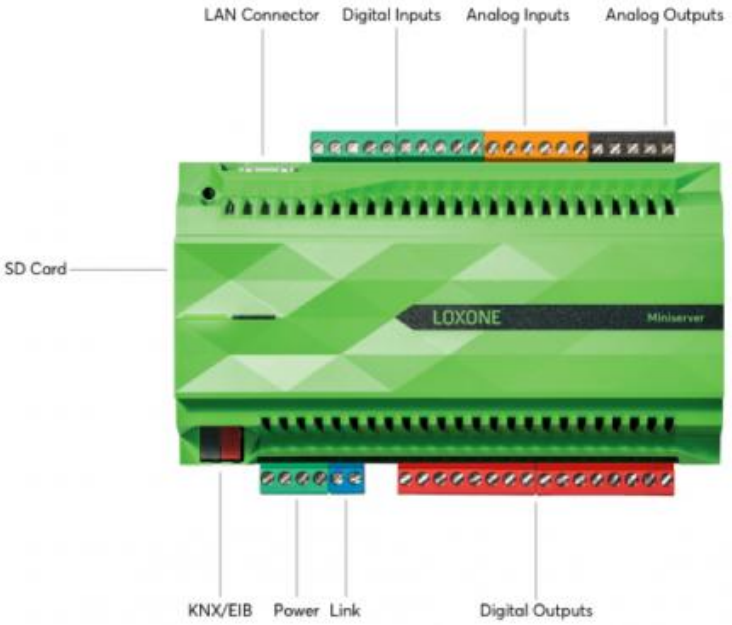

Ilustración 10. Miniserver Gen.1

A través del Loxone Link, el Miniserver puede conectarse con hasta 30 extensiones que le permiten ampliar el sistema y comunicarse con otros protocolos.

3.2.2 Touch tree

Es un pulsados capacitivo con 5 puntos de contacto, con un sensor de temperatura y otro de humedad integrado.

Se conecta mediante el cable Tree y se alimenta a 24 V.

Se representa en la ilustración 11.

Ilustración 11. Touch tree

3.2.3 Dimmer Tree RGBW y Tira Led

El Dimmer Tree RGBW es un dimmer para el control de la intensidad y el color de la tira Led. Diseñado para montarlo en un carril DIN, se alimenta a 24V y tiene 4 salidas PWM.

Se representa su conexionado en la ilustración 13.

La tira led se alimenta a 24V y ofrece cualquier color e intensidad deseada con un ángulo de haz de 120º y una temperatura de color de 3300K. Se pueden cortar cada 10cm o 6 LEDs. Se representa en la ilustración 12.

Ilustración 12. Tira Led

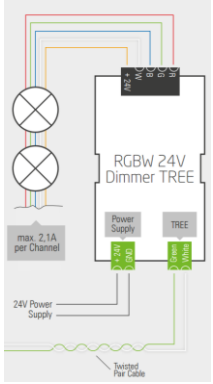

Ilustración 13. Dimmer Tree RGBW

3.2.4 Sensor de Movimiento y Luminosidad tree

Permite la detección del movimiento, presencia y luminosidad.

Cuenta con un ángulo de detección de 360° horizontalmente y 110° verticalmente, en un diámetro de 8 metros (con una altura de techo de 3 metros).

Dispone de sensor acústico ajustable para la detección de presencia.

Se representa en la ilustración 14.

Ilustración 14. Sensor de presencia

3.2.5 Contacto de Puerta/Ventana

Nos permite conocer el estado de las ventanas o puertas donde esté instalado.

Está formado por un contacto Reed, que se coloca en el marco de la ventana o de la puerta, y un imán colocado en la parte móvil.

Se conecta a una entrada digital del Miniserver.

Se representa en la ilustración 15.

Ilustración 15. Contacto de Puerta/Ventana

3.2.6 Extension Air

Se trata de una pasarela que recibe los mensajes enviados por los dispositivos Loxone Air y envía estos mensajes al Miniserver a través de la conexión Link que tiene con él, a la vez, recibe del Miniserver mensajes que tiene que enviar al dispositivo Air correspondiente.

Soporta un total de 128 dispositivos y se comunica con los dispositivos por la frecuencia 868MHz (EU).

Se instala en un carril DIN. Se representa en la ilustración 16.

Ilustración 16. Extension Air

3.2.7 Contacto de Puerta/Ventana Air

Al igual que el anterior contacto de Puerta/Ventana nos permite saber el estado de la puerta o ventana donde este instalado, pero comunica esta información vía Air, por la frecuencia 868MHz (SRD Band Europe), al Miniserver. Se alimenta con una pila botón CR2032. Se representa en la ilustración 17.

Ilustración 17. Contacto Puerta/Ventana Air

3.3 Instalación de los dispositivos Loxone en OpenHAB 3

Estos son los pasos seguidos para la instalación de los dispositivos Loxone en OpenHAB 3.

1. Se selecciona en el menú de la izquierda "Configuración" y aparece la pantalla representada en la ilustración 18 donde se selecciona "Bindings".

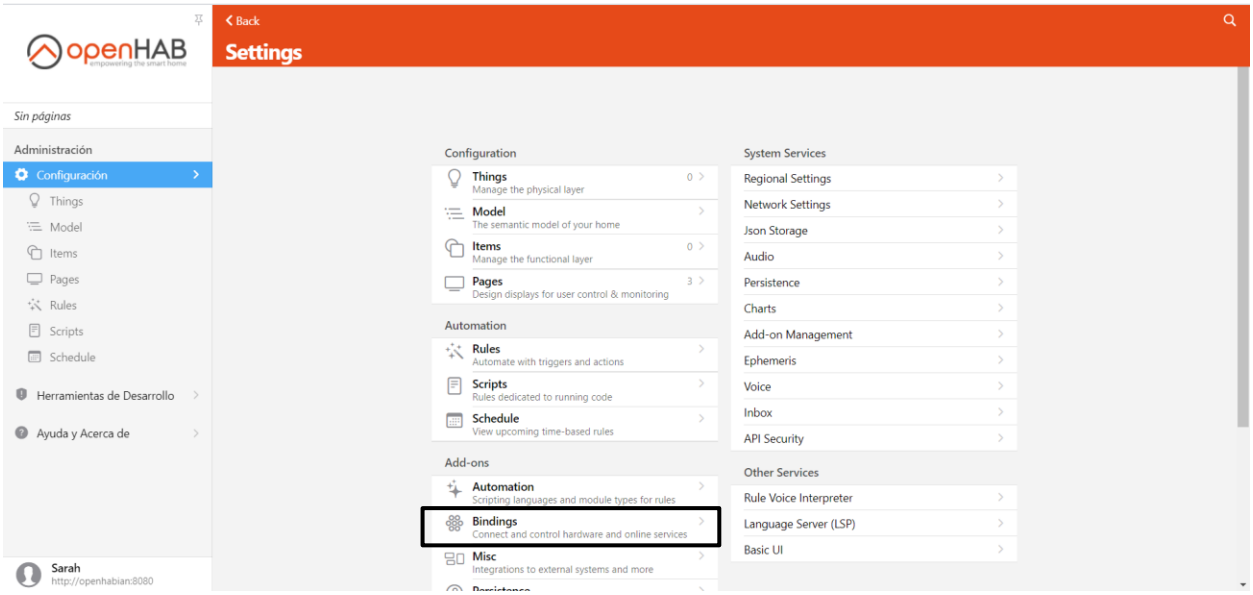

Ilustración 18. Pantalla de Configuración

2. Una vez dentro de Bindings (ver ilustración 19), se pulsa en el botón de azul situado abajo a la derecha para añadir un nuevo add-ons.

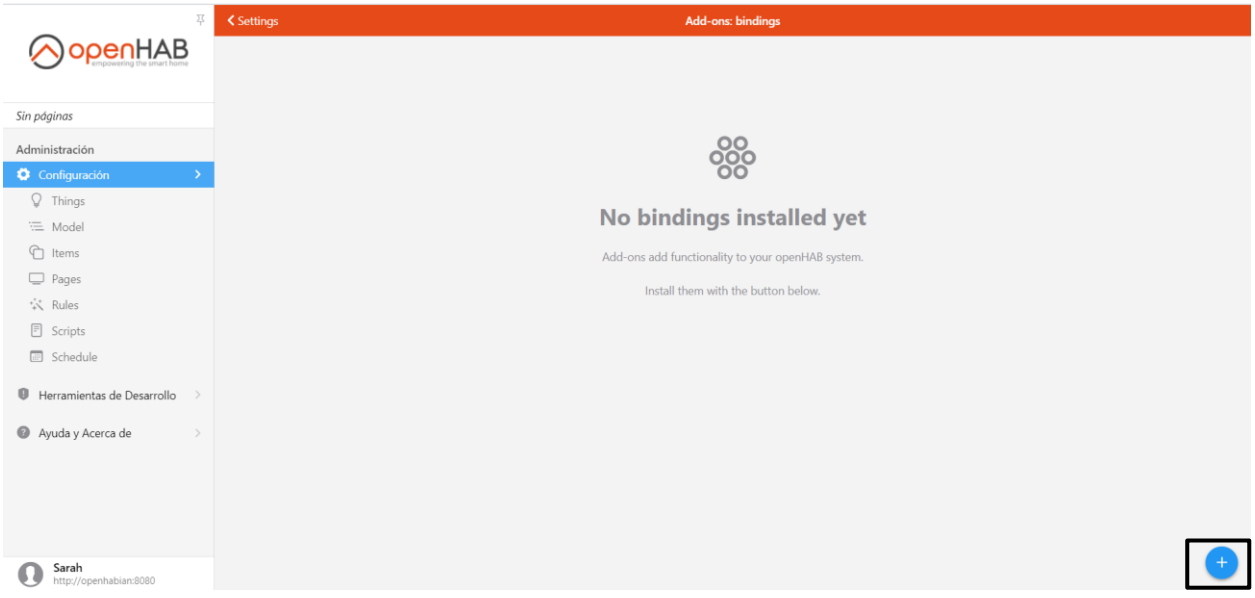

Ilustración 19. Pantalla de Bindings instalados

3. Aparecerá una pantalla con una lista de todos los binding add-ons disponibles, se busca el de Loxone y se selecciona “Install” para instalarlo, tal como aparece en la ilustración 20.

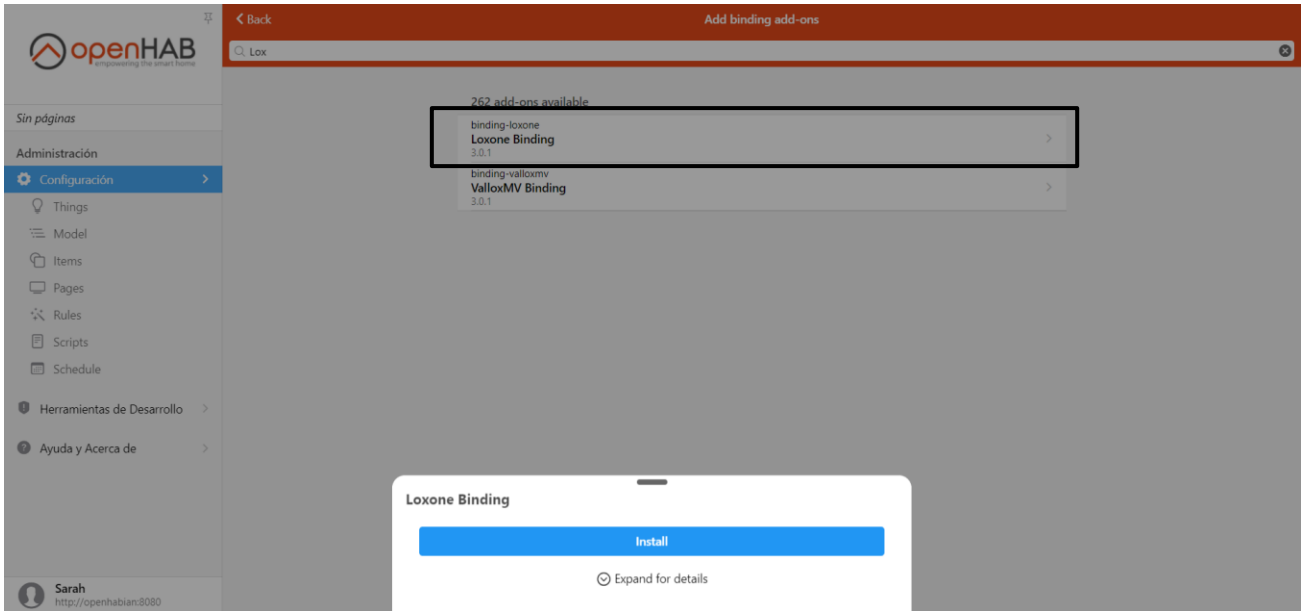

Ilustración 20. Instalación Binding de Loxone

4. Una vez instalado aparece la pantalla representada por la ilustración 21, donde queda confirmada su instalación.

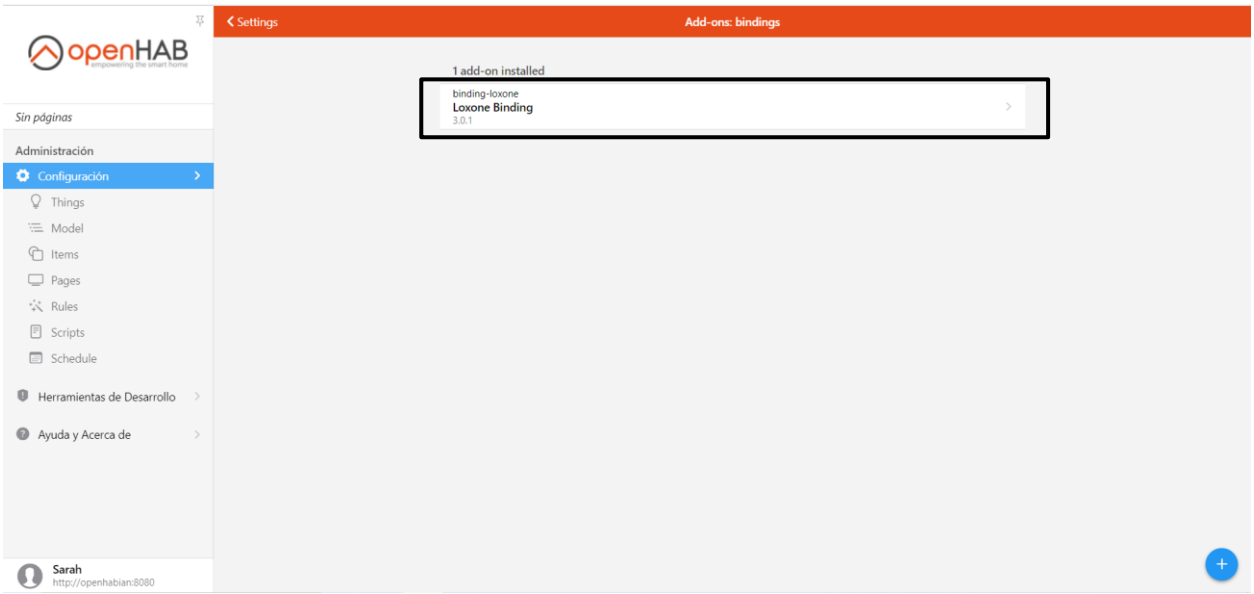

Ilustración 21. Pantalla de Binding con Loxone instalado

5. Posteriormente se selecciona el apartado “Things” del menú de la izquierda (aparecerá la pantalla representada en la ilustración 22) para lograr en primer lugar la conexión de OpenHAB con el Miniserver de Loxone.

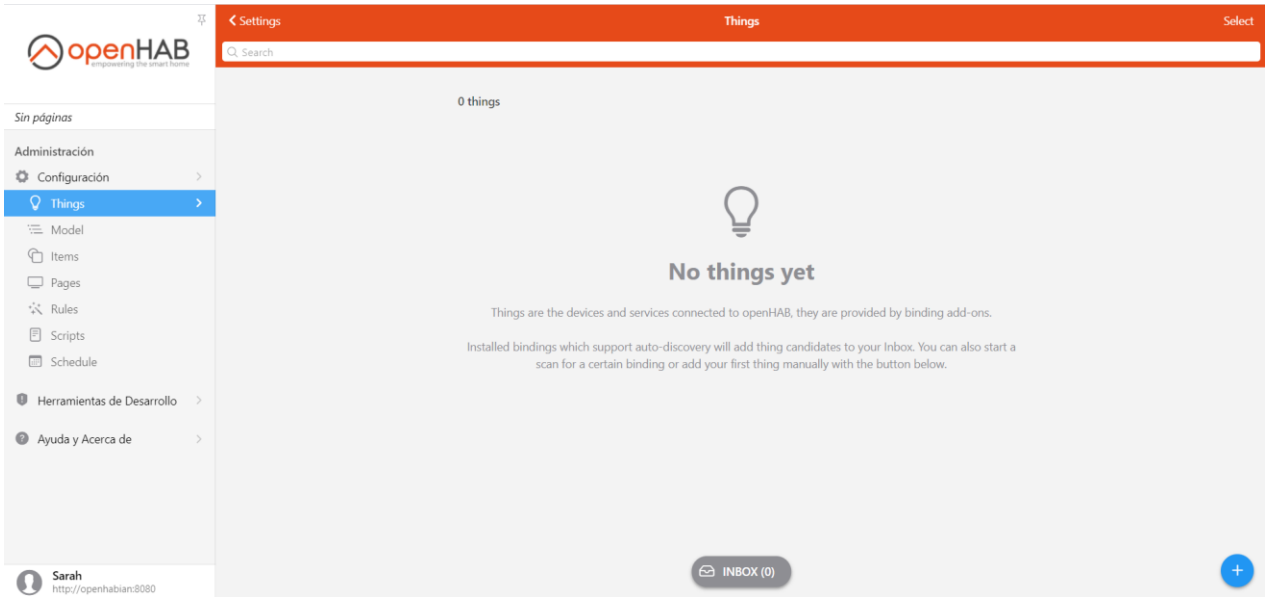

Ilustración 22. Pantalla de Things instalados

6. Se pulsa en el botón azul situado abajo a la derecha y aparece la siguiente página, representada en la ilustración 23, en la que se pulsa sobre “Loxone Binging”.

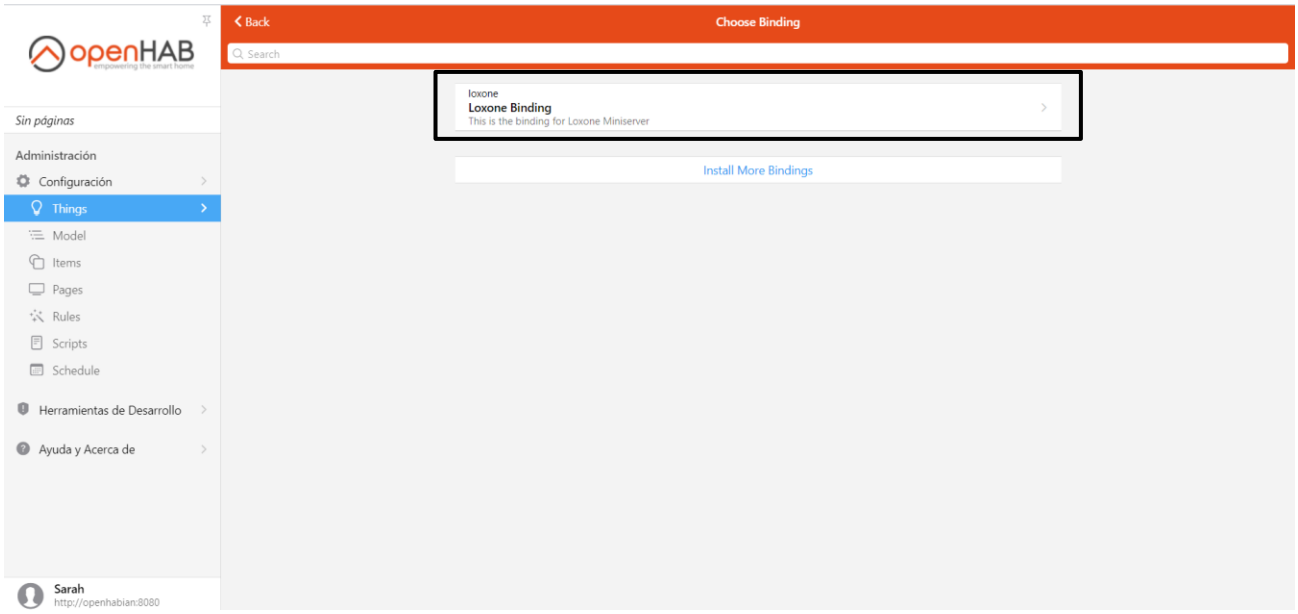

Ilustración 23. Elección del binding para instalar un Thing

7. Según se ve en la ilustración 24, se puede obtener de dos formas la comunicación con el Miniserver, ya sea pulsando "Scan" para escanear la red y lograr la comunicación de forma automática o añadiendo el Miniserver de forma manual.

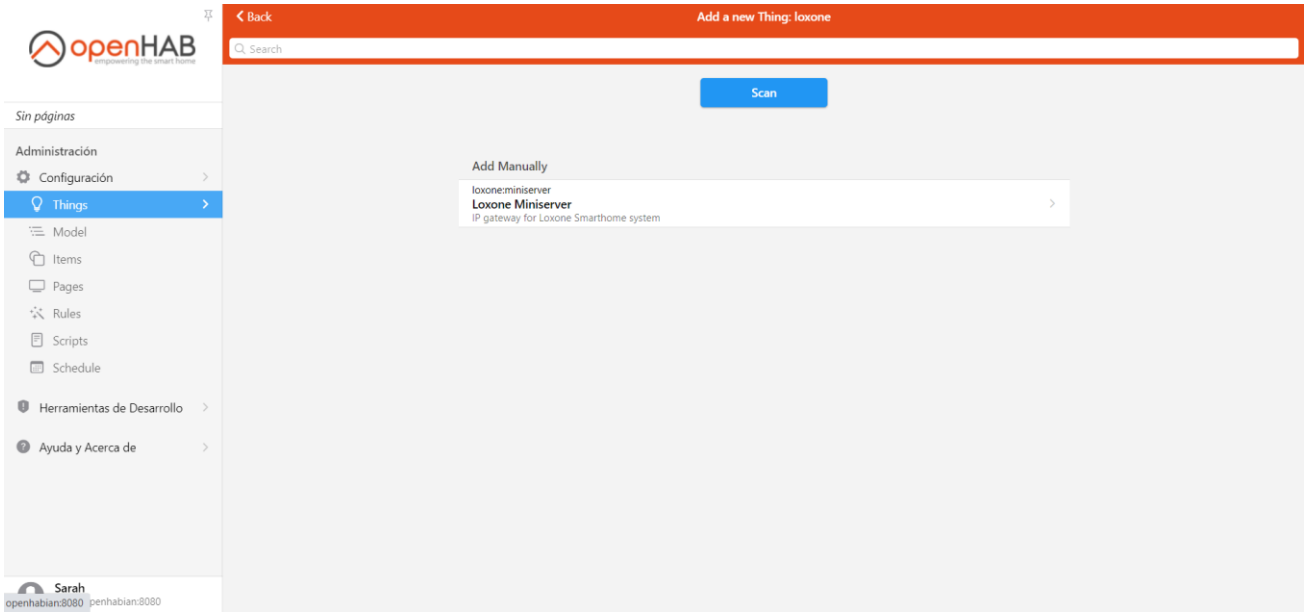

Ilustración 24. Pantalla para añadir un Thing de Loxone

8. Usando el formato manual aparece la pantalla representada por la Ilustraciones 25 y 26, y hay que rellenar los siguientes campos:

   i) *Unique ID:* Se escribe el número de serie del Miniserver.
   ii) *Host:* Se escribe la IP del Miniserver.

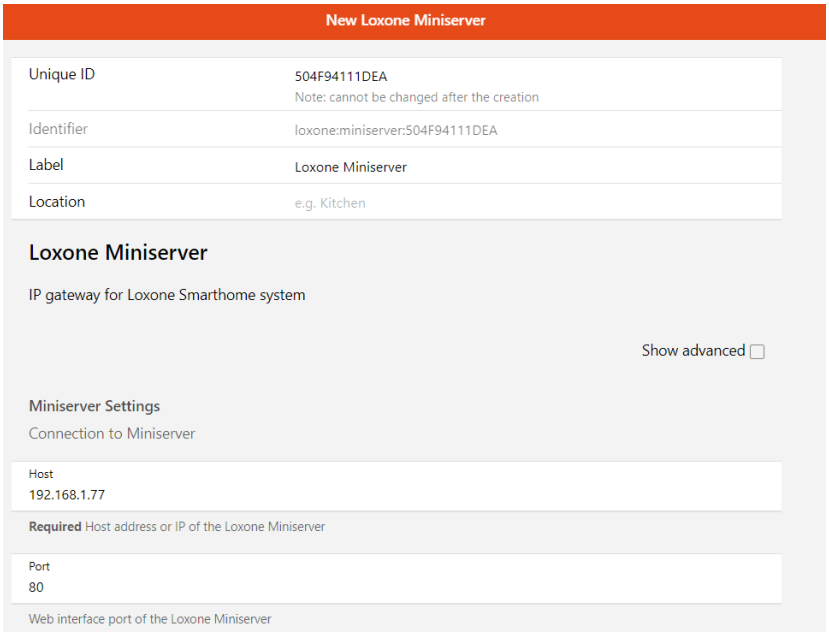

Ilustración 25. Configuración de Loxone Miniserver 1

iii) *User:* Se escribe el nombre de usuario con el que acceso al Miniserver.
iv) *Password:* Se escribe la contraseña de dicho usuario.

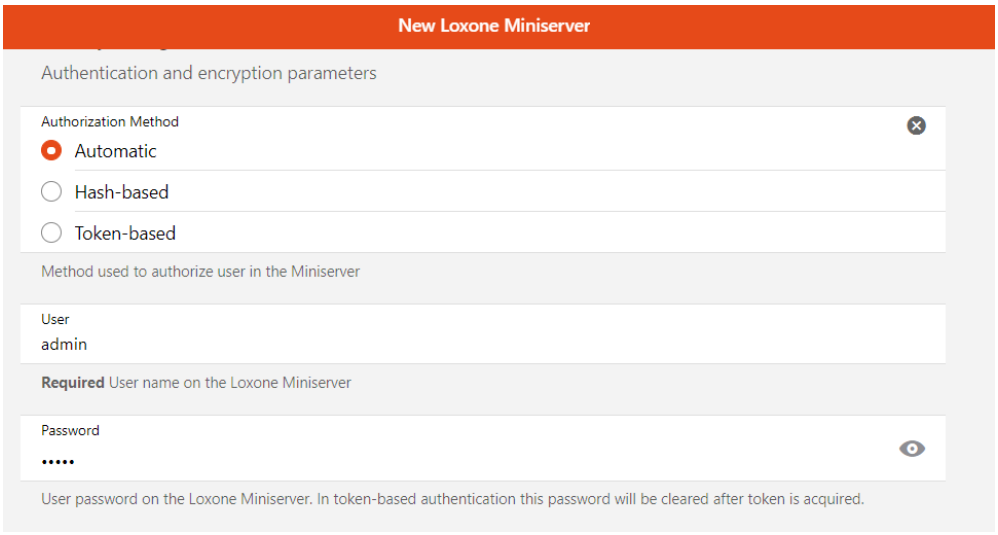

Ilustración 26. Configuración de Loxone Miniserver 2

9. Se pulsa el botón azul del final de la página para crear el Thing del Miniserver y aparece la pantalla representada en la ilustración 27, en la que se ve el primer Thing creado.

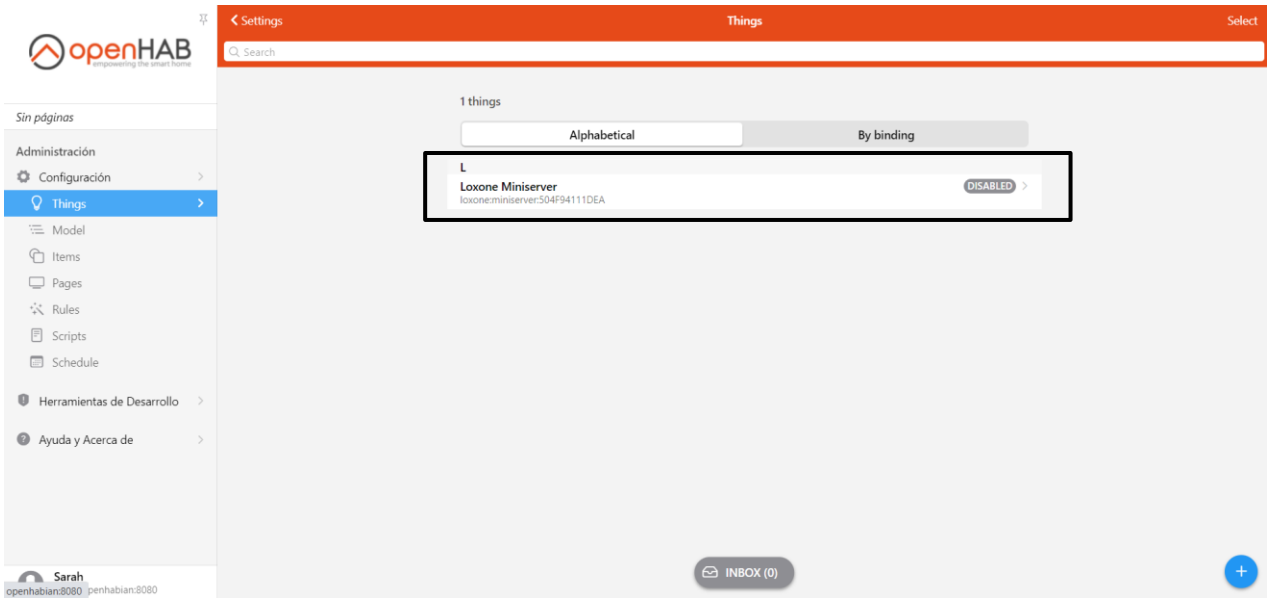

Ilustración 27. Pantalla de Things instalados con el Miniserver de Loxone

10. Al crearlo, su “Status” aparece como DISABLED. Para que se conecte hay que pulsar sobre Loxone Miniserver y en la siguiente página, representada por la ilustración 28, se pulsa sobre el icono de pausa.

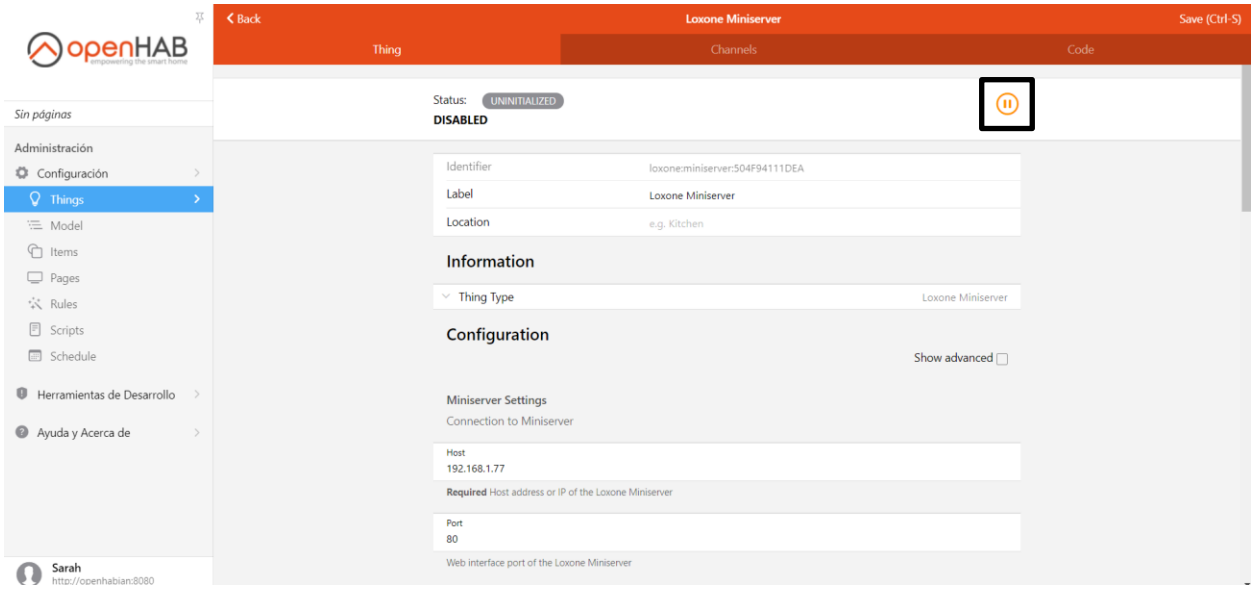

Ilustración 28. Activación del Thing del Miniserver

El “Status” pasa a ser ONLINE, como se representa en la ilustración 29, por lo que la conexión entre openHAB 3 y el Miniserver está lista y ya se puede comenzar a crear Links entre los Channels que nos ofrece el Miniserver y los Item que queramos tener para poder hacer uso de estos Channels.

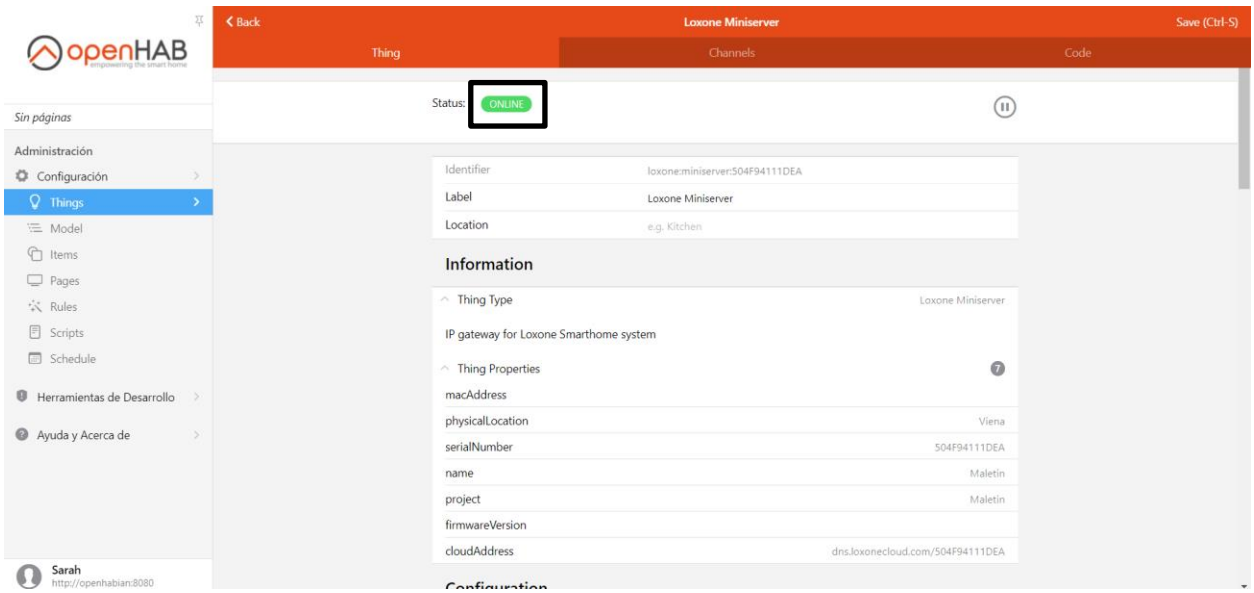

Ilustración 29. Estado Online del Thing del Miniserver

11. Al pulsar en la pestaña del medio llamada “Channels” se ven todas las funcionalidades ofrecidas por el Miniserver que se pueden controlar mediante Items (ver ilustración 30). Para poder usarlos hay que pulsar sobre el que se quiera agregar y seleccionar “Add Link to Item…”.

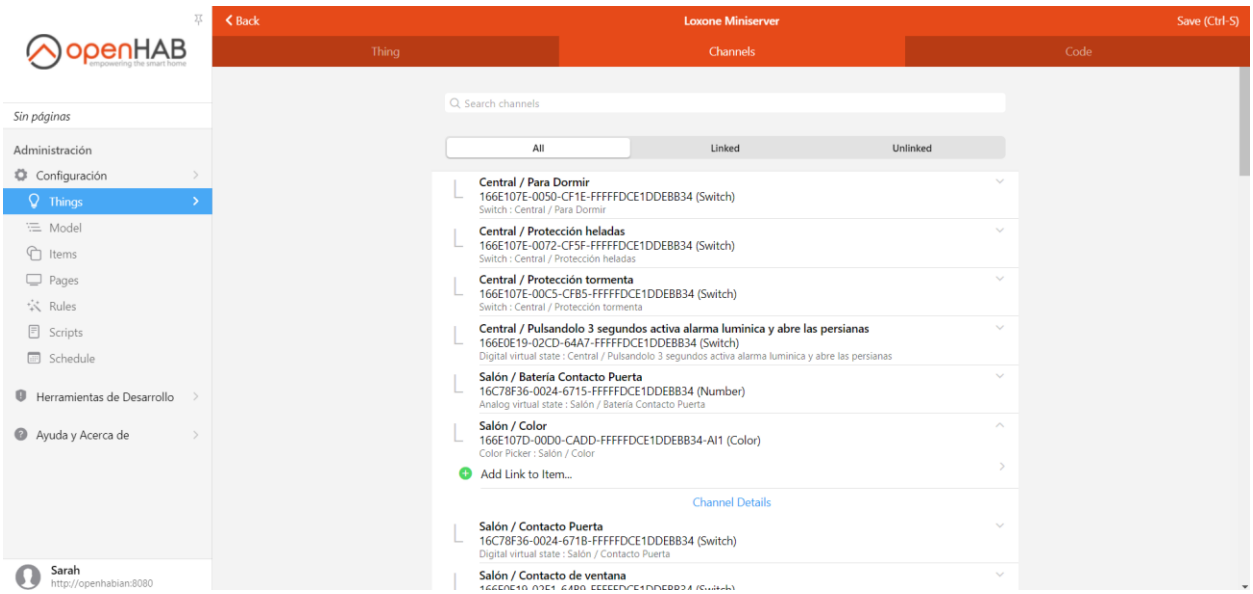

Ilustración 30. Channels del Miniserver

Aparece una pantalla, representada en la ilustración 31, en la que se marca “Create a new Item” y se rellenan los siguientes campos:

- *Name:* Un nombre cualquiera.
- *Label:* Se utiliza para describir un Item de una manera legible.
- *Type:* Mencionados en el punto 2.2.4 Items.
- *Category:* Para seleccionar el icono asociado al Item.

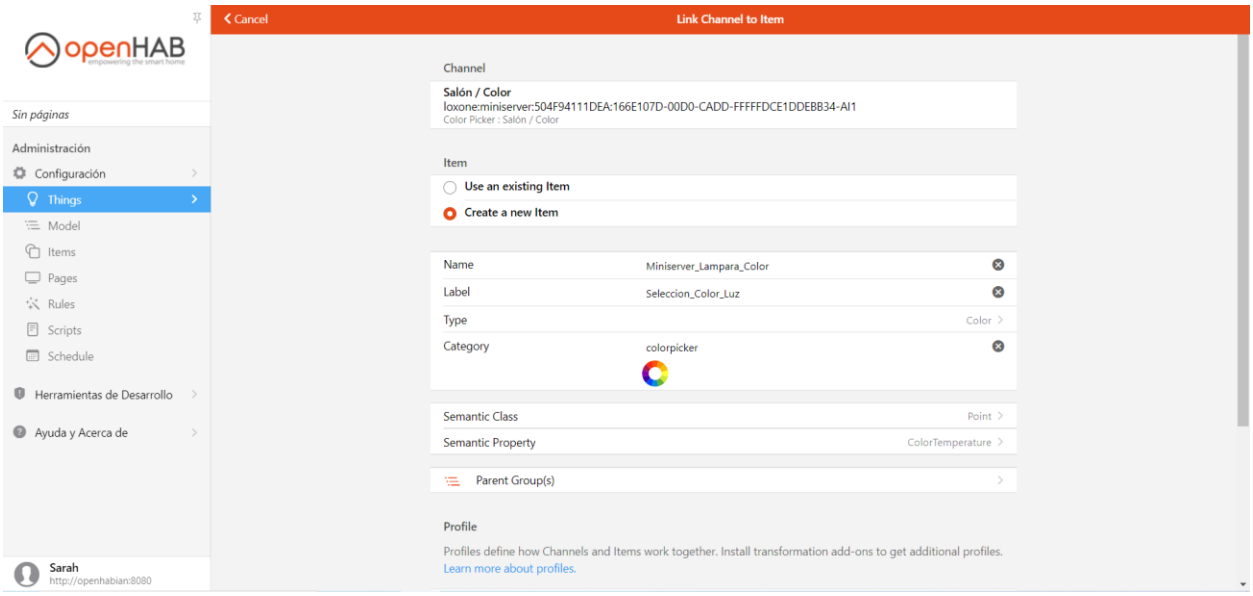

Ilustración 31. Creación de un Item

Finalmente, se pulsa sobre el botón azul “Link” situado al final de la página y aparece el Channel asociado al Item creado, representado en la ilustración 32.

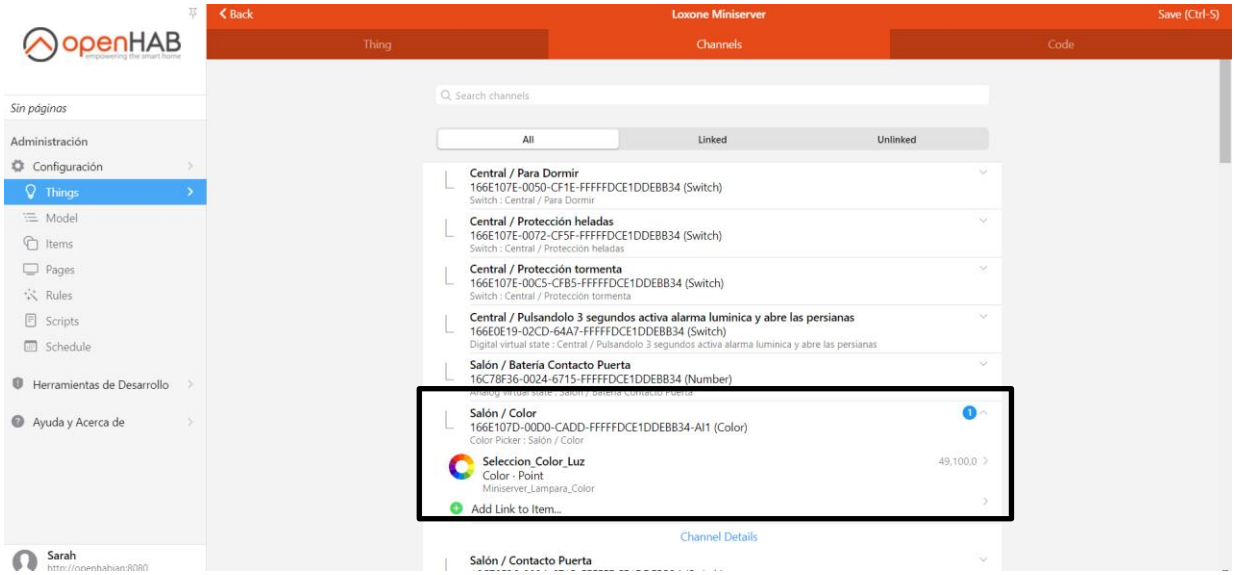

Ilustración 32. Channel con un Item asociado

12. Al pulsar sobre “Items” en el menú de la izquierda, aparece el Item creado y al pulsar sobre él da la posibilidad de modificar sus valores como se ve en la ilustración 33. En este caso, permite seleccionar el color, la intensidad y la saturación de los Leds de Loxone.

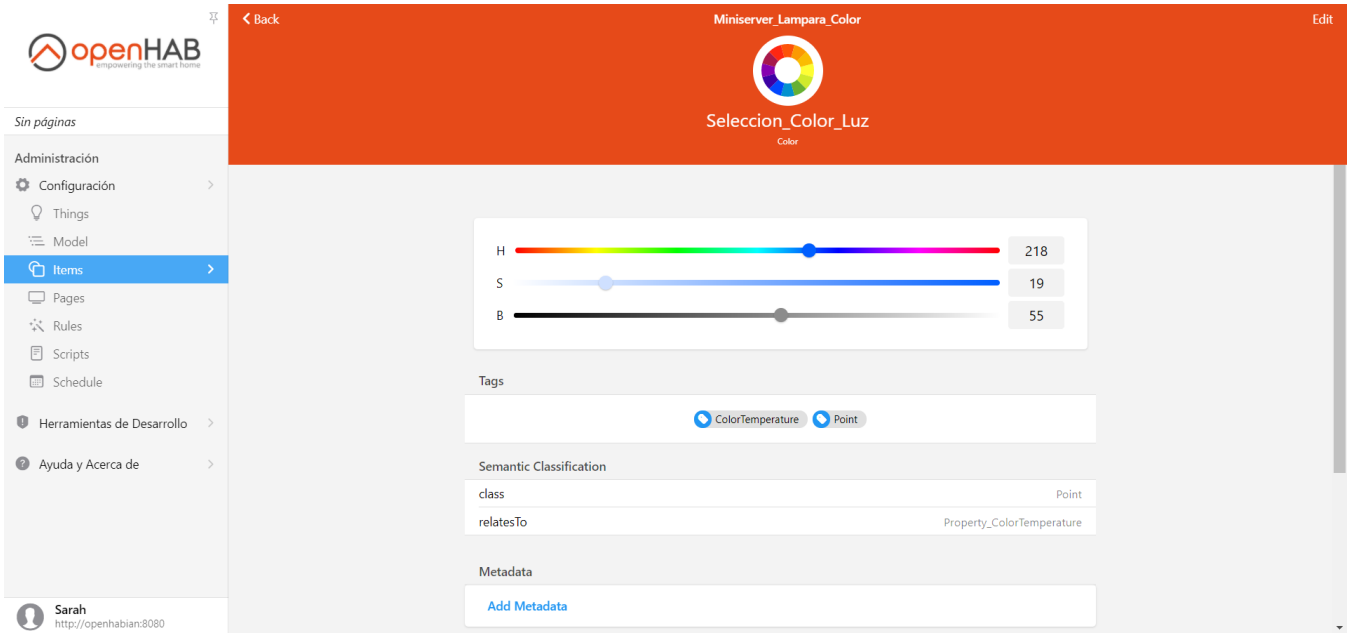

Ilustración 33. Item Lámpara de Color

13. Una vez realizados los Link con los Channels deseados se obtienen los Items representados en la ilustración 34, con los que se pueden controlar los dispositivos Loxone:

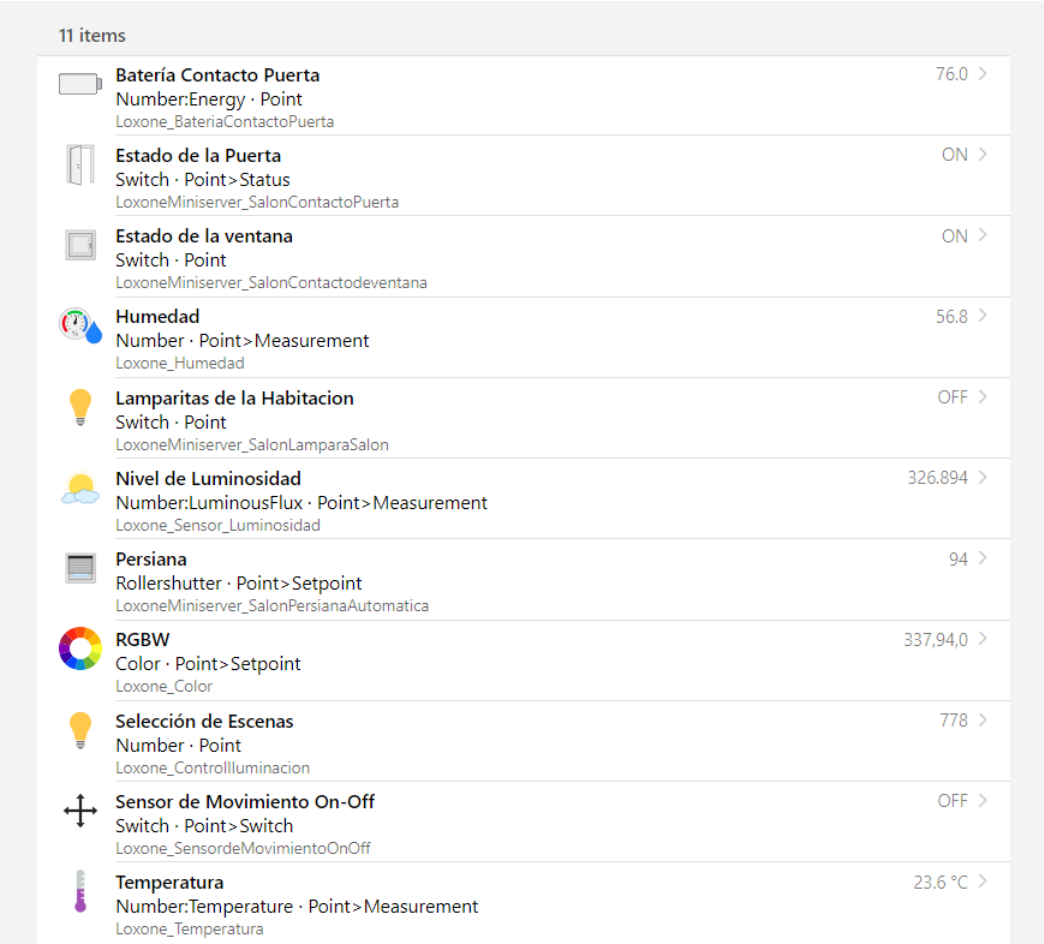

Ilustración 34. Items Loxone

4 Z-WAVE

En este capítulo se describe el protocolo Z-wave y se explican conceptos fundamentales del mismo. Posteriormente se describen los dispositivos usados para controlarlos mediante openHAB 3, así como los pasos seguidos para la vinculación de dichos dispositivos con openHAB 3.

4.1 Introducción

Z-wave [3] es un protocolo de comunicación inalámbrica específico para la automatización de la vivienda.

Comenzó siendo un protocolo propio hasta que en 2012 fue ratificado por la ITU y pasó a ser un estándar abierto. Los dispositivos de los diferentes fabricantes deben pasar una prueba de certificación que verifica que la capa de aplicación cumple con el estándar.

Algunas de las características a destacar de los dispositivos Z-wave son:

- Es un sistema fiable de topología mallada que admite hasta 232 dispositivos y 20 metros entre dispositivos en espacios cerrados.
- Sus dispositivos usan poca energía, ya que pueden operar en modo ahorro de batería, permaneciendo activos solo un 0.1% del tiempo. Esto permite crear dispositivos que se alimenten mediante pequeñas baterías (3,3V) durante 2 o 3 años.
- Usa la banda de 800-900 MHz, muy alejada de las redes como la WiFi o 5G.
- El sistema es accesible desde cualquier lugar haciendo uso de un ordenador, tablet o smartphone gracias a las aplicaciones desarrolladas por terceros.
- Es un sistema escalable por lo que se puede ir creando poco a poco sin necesidad de hacer una fuerte inversión inicial.
- Es un sistema interoperable gracias a la multitud de fabricantes homologados, lo que facilita la obtención fácil de dispositivos y un coste menor.

El protocolo Z-wave se divide en tres capas:

1) Capa de Radio: En esta capa se define la manera de intercambiar señales entre trasmisor y receptor.

   Se establece que con el incremento de la frecuencia de la portadora se representa un "1" lógico y su decremento es un "0" lógico. Usa el código Manchester con modulación GFSK o por desplazamiento en frecuencia gaussiana y puede funcionar a 9600bps o 40Kbps.

2) Capa de Red: Define el intercambio de datos de control. Se subdivida en tres capas:
   - *MAC:* Gestiona el hardware de los dispositivos inalámbricos para controlar el medio radio mediante el mecanismo de evitación de colisiones.
   - *Transporte:* Controla la trasferencia de mensajes entre nodos. Dichos nodos son identificados de manera inequívoca por los parámetros Home ID, encargado de identificar una red Z-wave, y Node ID, encargado de identificar a cada nodo de la red.
   - *Enrutado:* Controla el encaminamiento de las tramas que se envían entre los nodos.

Los nodos se dividen en dos tipos:

- *Esclavos:* Son nodos que reciben mensajes de un nodo controlador y responden a estos mensajes. Algunos esclavos incluso pueden enviar información no solicitada a otros nodos siempre que el controlador haya establecido este tipo de envíos.

- *Controladores:* Son dispositivos que conocen todas las rutas posibles entre los nodos esclavos y son los encargados de enviar los mensajes necesarios a los nodos esclavos. Estos controladores pueden ser estáticos al estar conectados a la red eléctrica o puede ser portátiles al ser alimentados por baterías y tendrán un estado de hibernación para prolongar la duración de las baterías. Estos nodos son los encargados de la inclusión y exclusión de los nodos esclavos en la red Z-wave. Si un nodo esclavo ya fue incluido en otra red distinta de la del controlador, este debe realizar el proceso de exclusión antes de hacer el de inclusión para dicho nodo. Es posible añadir controladores secundarios a una red Z-wave.

3) Capa de Aplicación: Define la comunicación entre los nodos.

   Todos los nodos deben soportar el comando "Basic" formado por el subcomando SET, para establecer valores, GET, para contestar con un valor y REPORT, para responder al GET. Aparte de estos comandos podrán tener más funciones dependiendo de la clase de dispositivos que sea.

   Todos los dispositivos vienen con valores predefinidos de fábrica para que puedan operar desde el mismo momento de incluirlos en la red Z-wave.

4.2 Dispositivos Z-wave

4.2.1 Z-Stick Gen5 de Aeotec

Gracias a este dispositivo podemos crear una red de dispositivos Z-wave, siendo este su controlador primario.

Para la inclusión de dispositivos se debe pulsar su botón y esperar a que la luz parpadee en color azul. La luz deja de parpadear cuando está en el proceso de inclusión del dispositivo y volverá a parpadea una vez el proceso acabe.

Para la exclusión de dispositivos se debe pulsar el botón hasta que el led parpadee en naranja. La luz deja de parpadear y se mantiene en color azul cuando está en el proceso de exclusión del dispositivo y volverá a parpadea en color naranja una vez el proceso acabe.

También permite que un procesador host controle hasta 232 dispositivos Z-wave utilizando el protocolo de tecnología Z-wave.

Proporciona actualizaciones de firmware inalámbricas para dispositivos Z-wave, las cuales se llevan a cabo a través de su puerto USB, que también sirve como puerto de carga para su batería interna.

Tiene un alcance inalámbrico de hasta 150 metros.

Proporciona un 250% más de ancho de banda de comunicación y permite una comunicación encriptada AES-128 bit.

Se representa en la ilustración 35.

Ilustración 35. Z-Stick Gen5 Aeotec

4.2.2 Motion Sensor de Fibaro

Ilustración 36. Motion Sensor Fibaro

Es un multisensor Z-wave que, además de detectar movimiento usando un sensor IR pasivo, puede medir la temperatura y la intensidad de la luz. También incorpora un acelerómetro que capta cambios de posiciones, así como si intentan abrir su carcasa.

Es un dispositivo alimentado por batería y diseñado para instalarse rápida y fácilmente en cualquier superficie gracias a su diámetro de 44mm.

Se representa en la ilustración 36.

4.2.3 Flood Sensor de Fibaro

Es un sensor de inundación y temperatura universal compatible con Z-wave, y puede ser alimentado a batería o VDC.

Ilustración 37. Flood Sensor Fibaro

Está diseñado para colocarse en el suelo o en la pared con una sonda del sensor de inundación extendida por un cable conectado.

Este dispositivo es capaz de flotar sobre del agua y puede seguir enviando una señal de alarma, aparte de hacer parpadear su luz LED y hacer sonar una alarma acústica que tiene integrada.

Está equipado con un sensor de inclinación que informa de la inclinación o movimiento al controlador.

Se representa en la ilustración 37.

4.3 Instalación de los dispositivos Z-wave en OpenHAB 3

Estos son los pasos seguidos para la instalación de los dispositivos Z-wave en OpenHAB 3.

1. Al igual que se hizo anteriormente con los dispositivos Loxone, lo primero es instalar el binding de Z-wave, como se representa en la ilustración 38.

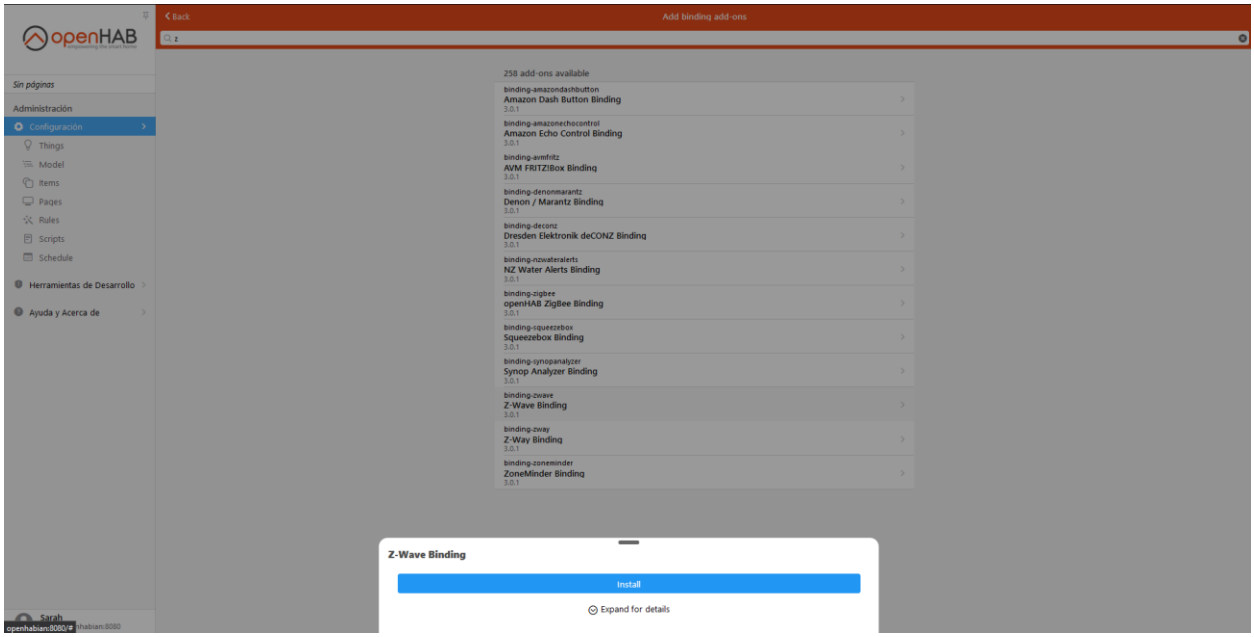

Ilustración 38. Instalación del binding de Z-wave

2. Una vez instalado el binding, desde el apartado "Thing" se crea un Thing nuevo seleccionando el binding de Z-wave instalado y aparece la pantalla representada en la ilustración 39.

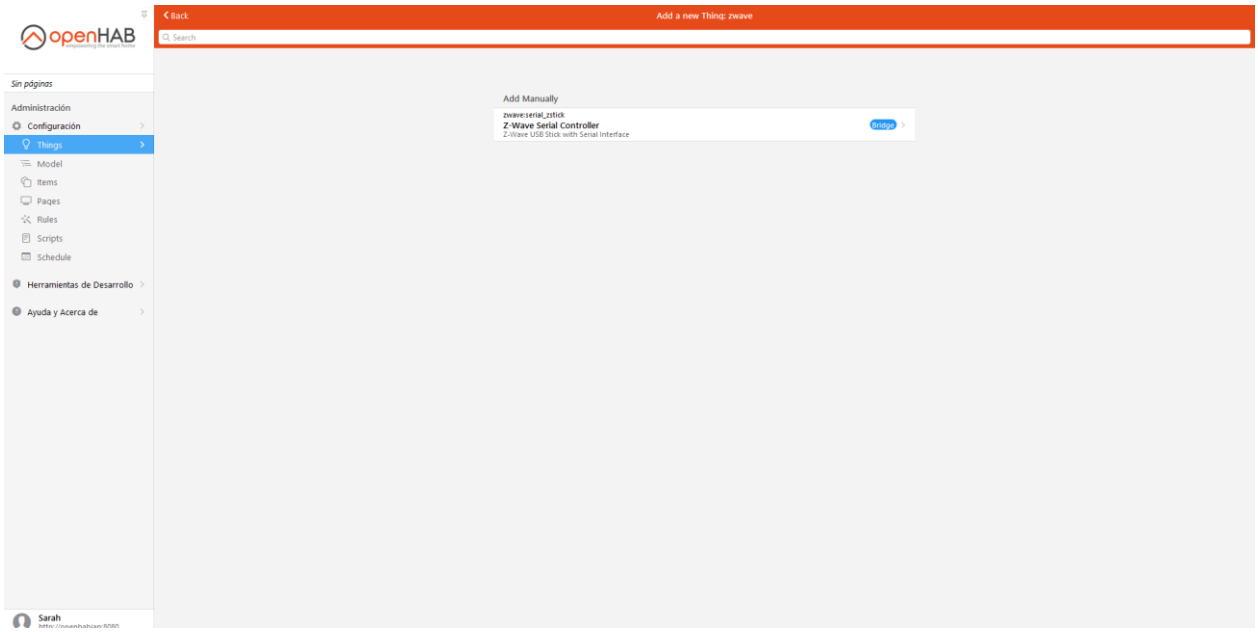

Ilustración 39. Creación del Thing del Controlador de Z-wave

Se selecciona el controlador de Z-wave que aparece gracias al stick Gen 5 insertado en la Raspberry Pi mediante un HUB de USB con alimentación externa. Esto ha sido necesario debido a un problema de hardware entre dicho stick y la RaspberryPi 4 que obliga a usar este tipo de dispositivo para que la RaspberryPi 4 reconozca el stick.

3. Una vez seleccionado el controlador Z-wave, aparece la pantalla representada por la ilustración 40, en la que se debe seleccionar el Serial Port asignado al stick y pulsar sobre “Create Thing”.

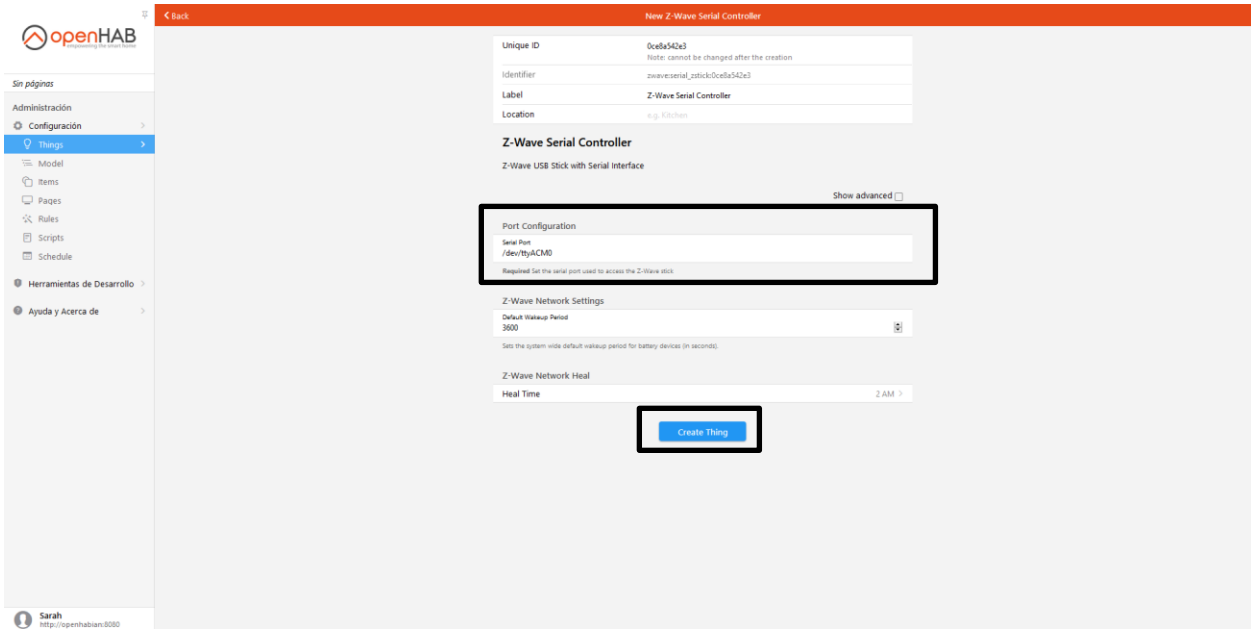

Ilustración 40. Selección del Serial Port del Stick

4. Unos segundos después aparece el controlador como “Online” y ya estará listo para añadir el resto de los dispositivos Z-wave. Para ello se pulsa sobre el apartado “Things” para seleccionar el Binding de Z-wave como aparece en la ilustración 41.

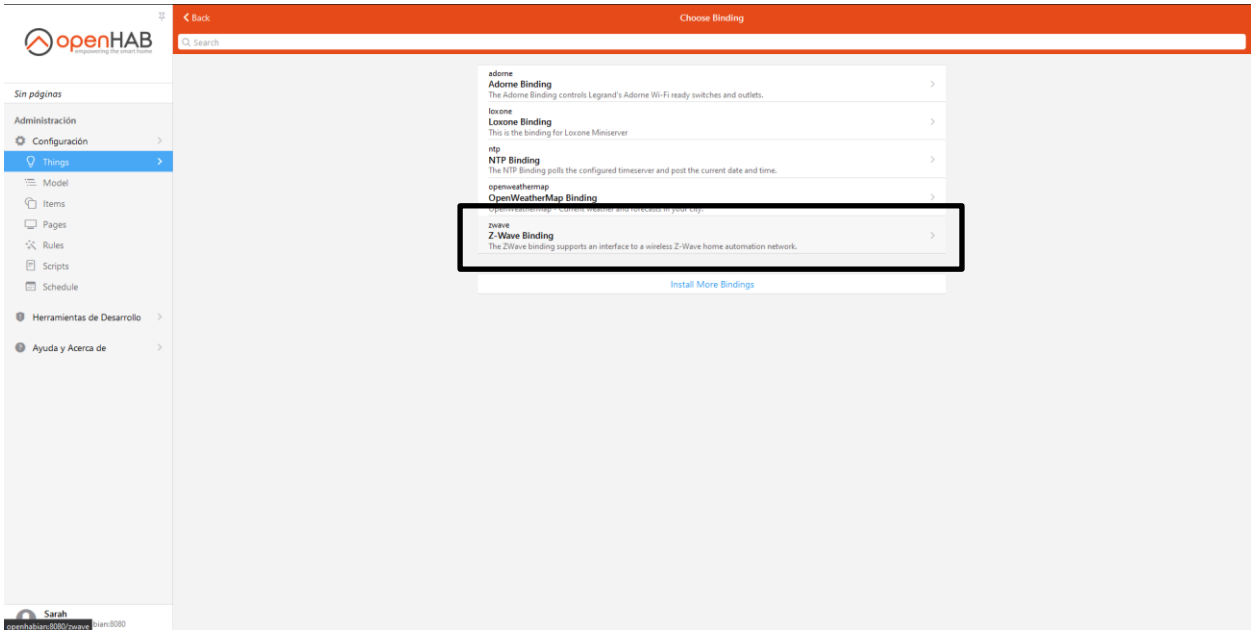

Ilustración 41. Thing del controlador de Z-wave instalado

5. A diferencia de lo realizado anteriormente con Loxone, es necesario crear un Thing por cada dispositivo Z-wave a integrar. Una vez seleccionado el binding de Z-wave hay que pulsar en escanear y esperar a que encuentre los dispositivos Z-wave ya configurados con el controlador instalado anteriormente. En la ilustración 42 aparece el dispositivo Motion Sensor de Z-wave.

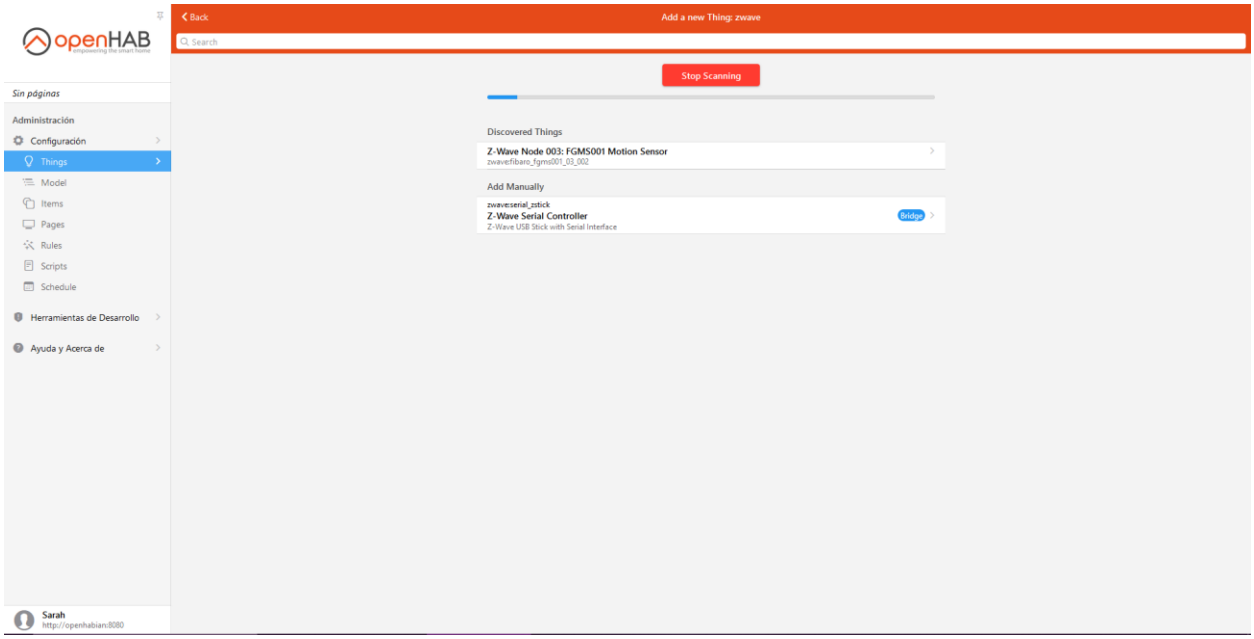

Ilustración 42. Búsqueda de dispositivos Z-wave

6. Tal y como aparece en la ilustración 43, al pulsar sobre el dispositivo que encontró nos da la opción de añadirlo como Thing.

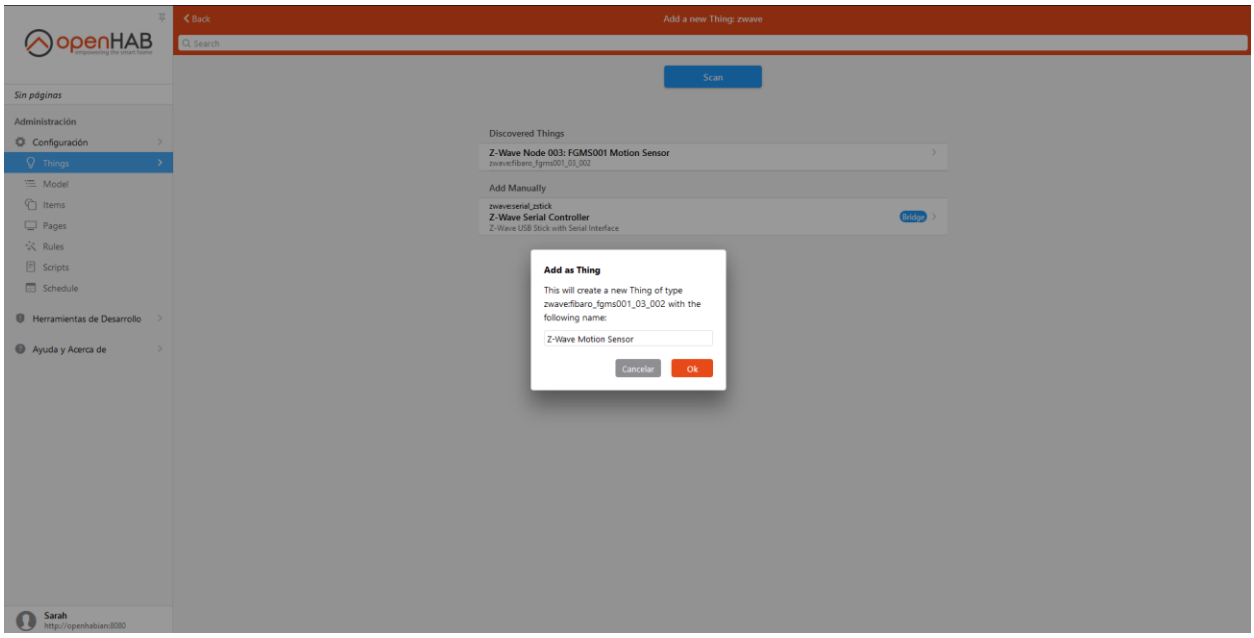

Ilustración 43. Añadiendo el Motion Sensor

7. Al pulsar sobre el Thing del dispositivo, en este caso, el sensor de movimiento, temperatura y luminosidad de Z-wave, se pueden configurar todos sus parámetros, al contrario que en los dispositivos Loxone. Algunos de estos parámetros están representados en la lustración 44.

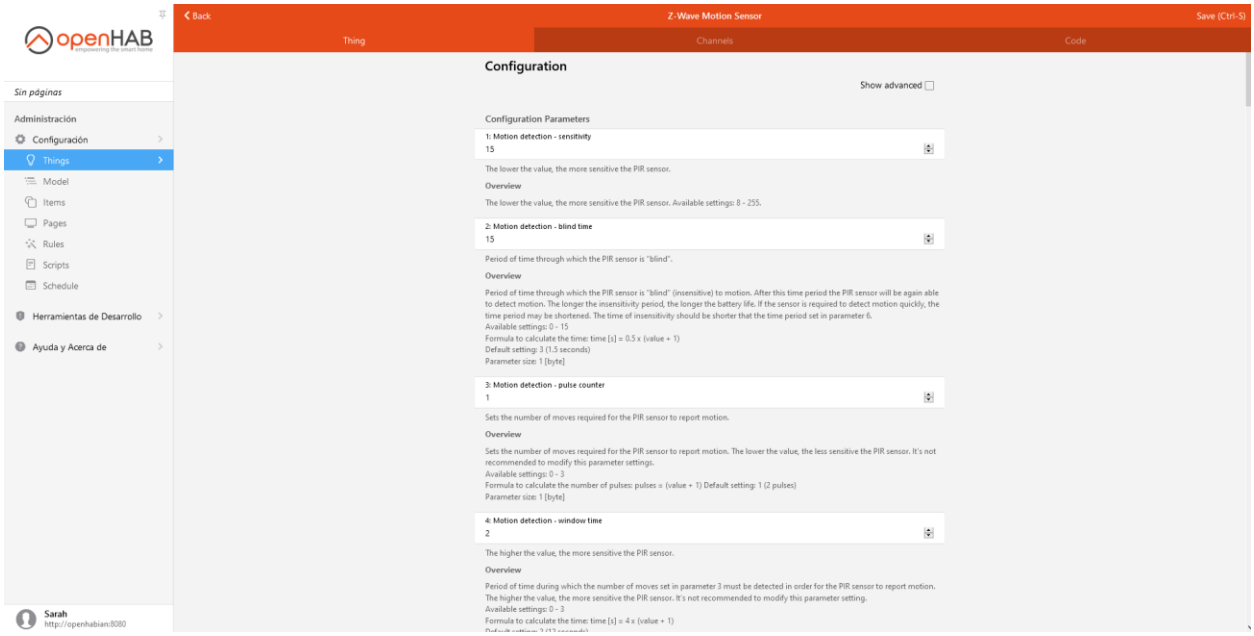

Ilustración 44. Configuración de parámetros del Motion Sensor

8. Pulsar en "Channel" para ver las funcionalidades que ofrece el dispositivo, representadas en la ilustración 45.

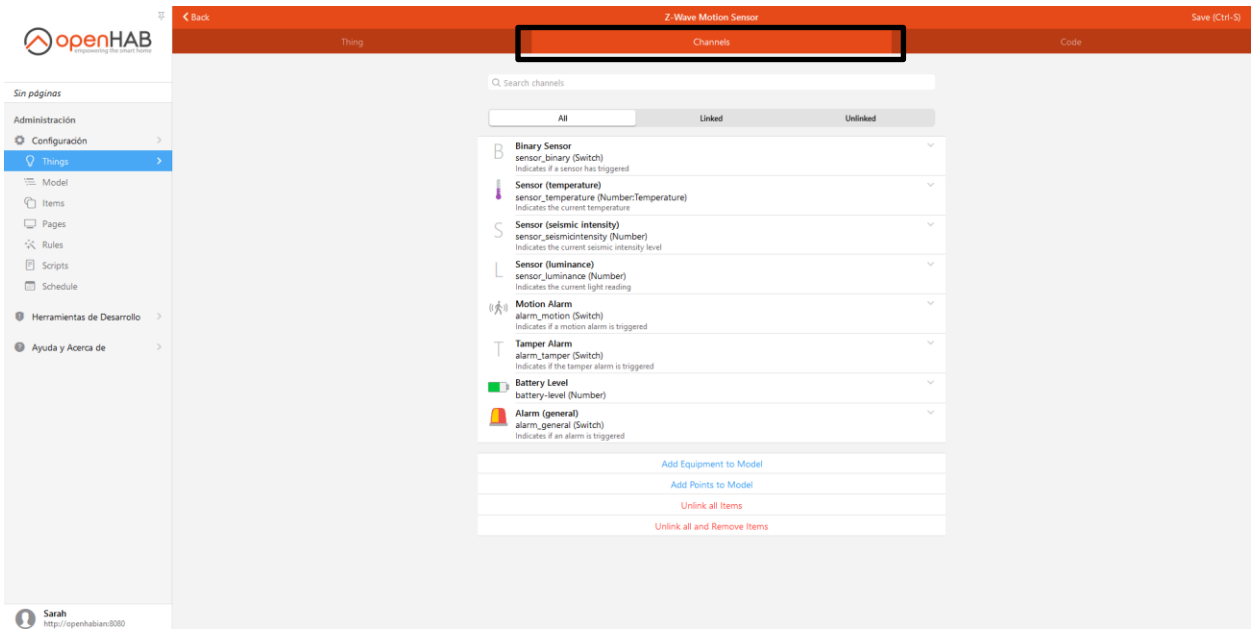

Ilustración 45. Channel del Motion Sensor

9. Al igual que con Loxone, se enlazan Items nuevos a cada "Channel" que se desee. En el siguiente ejemplo, representado en la ilustración 46, se crea un Item para el "Channel" que proporciona la información de la temperatura.

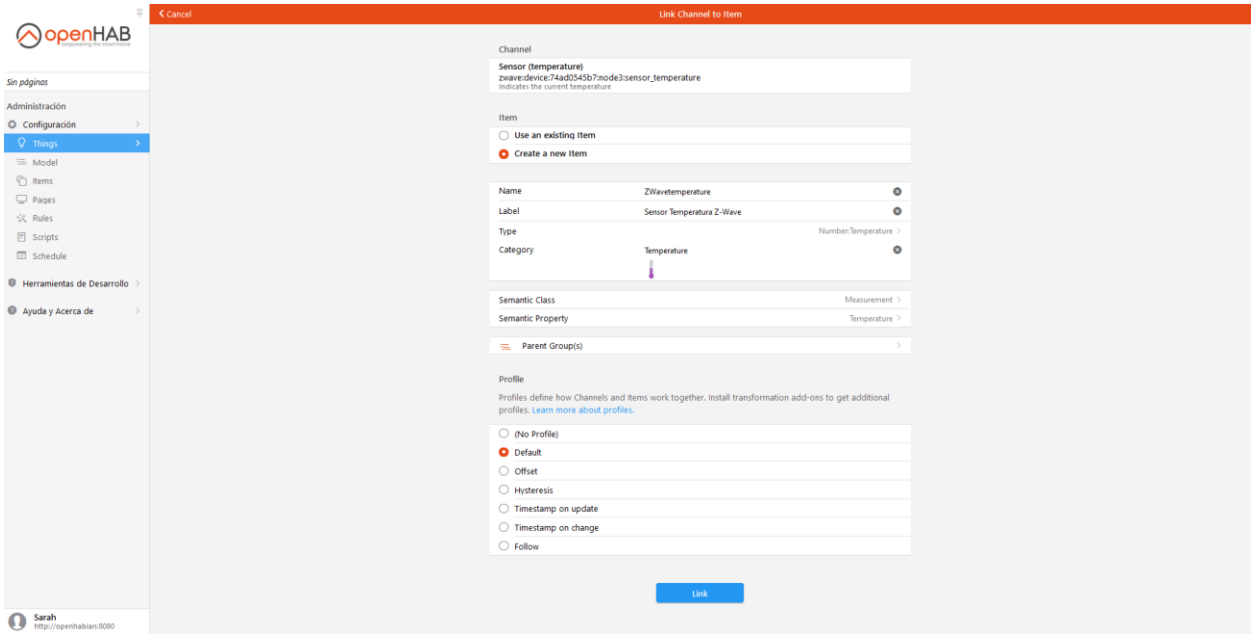

Ilustración 46. Creación del Item de Temperatura

10. En la ilustración 47 se representan los Items creados a partir de las funcionalidades que ofrecen los dispositivos Z-wave utilizados.

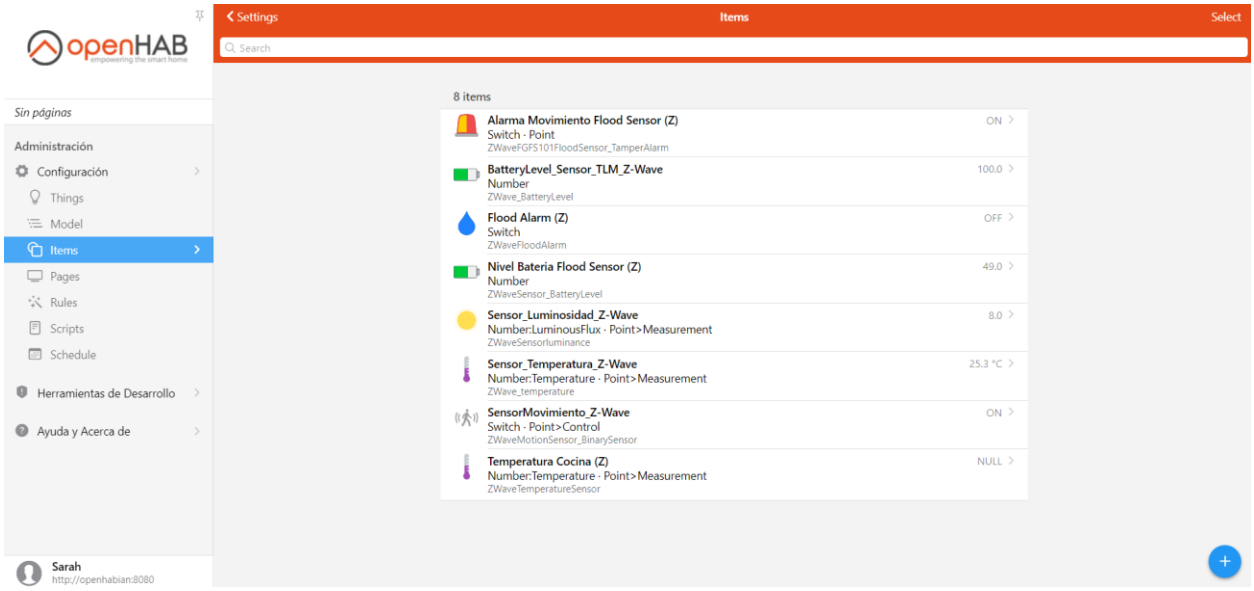

Ilustración 47. Items Z-wave

5 INTERFAZ DE USUARIO

En este capítulo se describirán las dos interfaces que se usarán para la visualización y manejo de los ítems creados en openHAB 3. Se describen los pasos seguidos para su creación.

5.1 Plano de la vivienda en 3D

5.1.1 HABpanel

HABpanel [4] es una interfaz de panel que permite crear interfaces desde el dispositivo destino de forma interactiva y sin necesidad de usar ningún Sitemap.

Se instala de forma predeterminada al instalar openHAB 3.

En la ilustración 48 se representan las entidades que componen HABpanel.

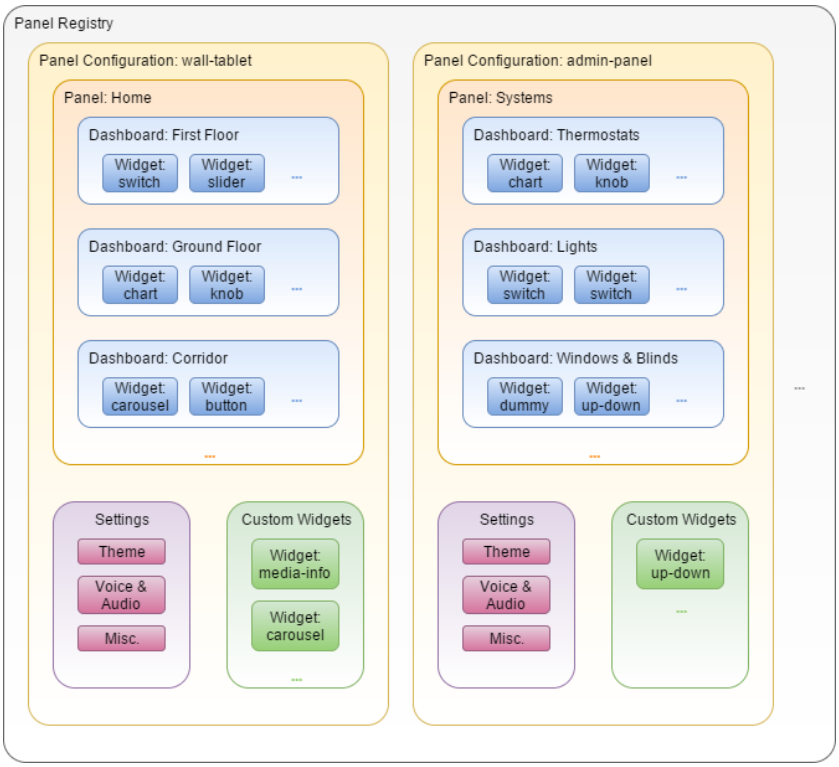

Ilustración 48. Terminología de entidades de HABpanel

- *Panel Registry*: Almacena todos los Panel Configuration.

- *Panel Configuration:* Contiene un Panel junto con su configuración y las definiciones de los widgets personalizados que se hayan creado en dicho Panel. Cada dispositivo en el que se ejecute HABpanel tiene una configuración de panel activada y muestra el Panel Configuration que está asociada a él.

- *Panel:* Está compuesto de Dashboards. Normalmente se usa un Panel por planta para tener una separación de la vivienda.
- *Dashboards:* Está compuesto por widgets. Se usa normalmente un Dashboards por estancia.
- *Widgets:* Son los componentes del Dashboards que permiten hacer uso de los ítems creados en openHAB o incluso servicios externos como Google Calendar. En la Tabla 4 se recogen los tipos de widgets que se pueden usar.

Tipo	Descripción
Button	Al pulsar sobre él realizará una acción, como enviar comandos a un Item o navegar a otro Dashboards. También puede ajustar sus colores según el estado del Item subyacente.
Chart	Puede aprovechar los servicios de persistencia de openHAB para trazar series numéricas durante un período de tiempo. También puede mostrar imágenes de gráficos generadas por el servidor.
Clock	Muestra un reloj analógico o digital. También se puede utilizar para mostrar la fecha actual.
Color Picker	Ofrece varias formas de mostrar y actualizar el estado de un Item (o grupo) de color.
Dummy	Muestra el estado actual de un elemento sin ninguna interactividad, junto con una etiqueta y un icono opcional.
Frame	Muestra una página web externa en HTML.
Slider	Refleja el estado y actualiza elementos numéricos dentro de un rango de valores. Hay varias opciones disponibles para modificar su apariencia y comportamiento.
Knob	Es similar en esencia al Slider, pero de forma giratoria. También ofrece una amplia capacidad de configuración sobre su apariencia y comportamiento.
Label	Muestra un texto fijo y tiene algunas opciones de apariencia (color, fuente).
Selection	Abre un menú o una cuadrícula de opciones configuradas automática o manualmente para enviar comandos a este elemento. Hay varias opciones de visualización disponibles.
Image	Puede mostrar una imagen y actualizarla a intervalos regulares.
Switch	Controla un Item de tipo Switch. Informa de su estado y puede alternarlo entre On y Off.
Timeline	Puede mostrar varios "carriles" de elementos con sectores codificados por colores que representan sus cambios de estado durante el período seleccionado. Muestran detalles sobre el estado del Item en un momento seleccionado.
Template	Permite renderizar y alojar una plantilla HTML AngularJS configurada por el usuario.

Tabla 4. Tipos de Widgets

Aparte de los widgets ya definidos podemos crear otros personalizados que podremos reutilizar, compartir y configurar. Se crean mediante una plantilla de AngularJS y podrán ser añadidos al Dashboards. La definición de estos widgets se almacena en el registro del Panel Configuration por lo que serán específicos de dicho Panel Configuration.

5.1.2 Sweet Home 3D

Sweet Home 3D [5] es un software de código abierto para realizar diseño de interiores que permite diseñar planos de casas y organizar muebles, cuadros, ventanas, etc. Realiza modelos virtuales en un plano 2D y ofrece una vista previa en 3D.

La aplicación tiene una gran variedad de elementos prediseñados y permite importar modelos creados por usuarios o por uno mismo, siempre y cuando estos modelos sean creados mediante un software capaz de generar ficheros en formato OBJ, DAE / Collada, KMZ o 3DS.

Este software está disponible en múltiples idiomas y puede ser ejecutado en Windows, Mac OS X 10.4 / 10.12, Gnu/Linux y Solaris mediante su instalación o usar su versión online sin necesidad de instalarlo.

Para este proyecto se usa un modelo de ejemplo de la galería de Sweet Home 3D, ya que diseñar el plano de una vivienda no es uno de los objetivos del proyecto.

5.2 Creación de la interfaz en 3D

En este apartado se describen los pasos seguidos para usar un plano 3D creado con Sweet Home 3D en OpenHAB 3 y usar sus elementos para interactuar con los Items creados.

5.2.1 Instalación del plano de Sweet Home 3D en HABpanel

Los pasos seguidos para esta instalación son los siguientes:

1. Instalación de Samba, como se representa en la ilustración 49, desde la herramienta de configuración de OpenHABian que aparece al ejecutar “sudo openhabian-config” en la consola.

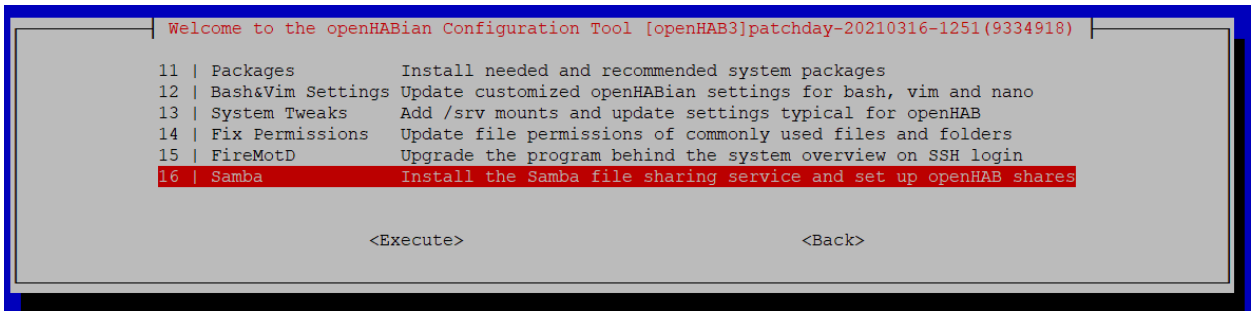

Ilustración 49. Activación de Samba

2. Descarga de Sweet Home 3D JS Viewer [8] y su extracción a una subcarpeta denominada “sweethome3d” dentro de la carpeta “conf/html” de la instancia de openHAB 3, ubicada en la instalación de OpenHABian en la Raspberry Pi, accediendo a la misma gracias a la activación de Samba.

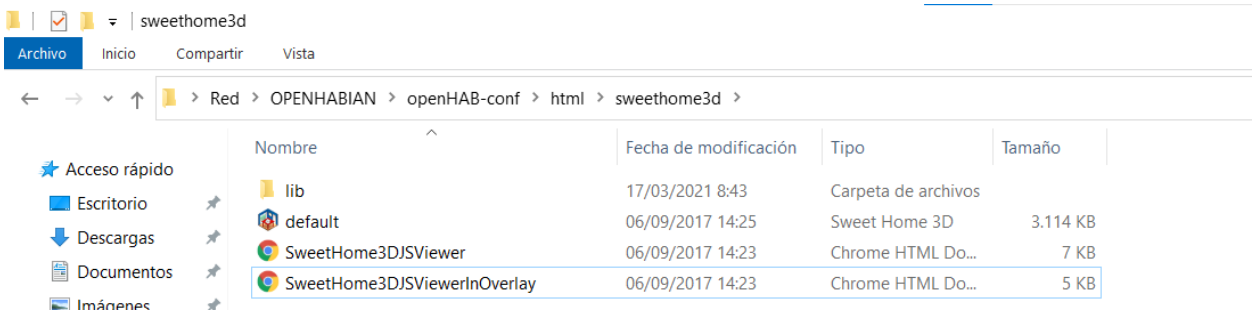

Ilustración 50. Contenido de la carpeta "sweethome3d"

El contenido de esta carpeta está representado por la ilustración 50 y es el siguiente:

- /sweethome3d/lib - contiene las bibliotecas de JavaScript adicionales para mostrar la vista WebGL.
- /sweethome3d/default.sh3d - contenido del diseño.
- /sweethome3d/SweetHome3DJSViewer.html - página de prueba.
- /sweethome3d/SweetHome3DJSViewerInOverlay.html - página de prueba con una superposición.

3. Descargar los archivos "*sweethome3d.directive.js*" [9] y *"3d-view.tpl.html"* [10] y colocarlos en "conf/html".

4. Colocar el archivo.sh3d de la vivienda a controlar en "conf/html", como se representa en la ilustración 51.

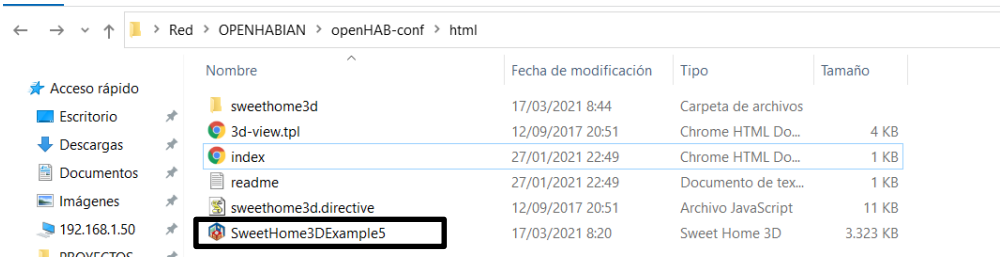

Ilustración 51. Colocar el plano en 3D en la carpeta correspondiente

5. Abrir HABpanel, crear un dashboard y agregar un widget del tipo Template (Modelo) tal y como se representa en las ilustraciones 52,53 y 54.

Ilustración 52. Pantalla inicial de HABpanel

Ilustración 53. Dashboard para la interfaz 3D

Ilustración 54. Agregación del widget Modelo

6. Editar este widget (como aparece en la ilustración 55), elegir la opción de "importar desde archivo", referenciada en la ilustración 56, y seleccionar el archivo *"3d-view.tpl.html"*.

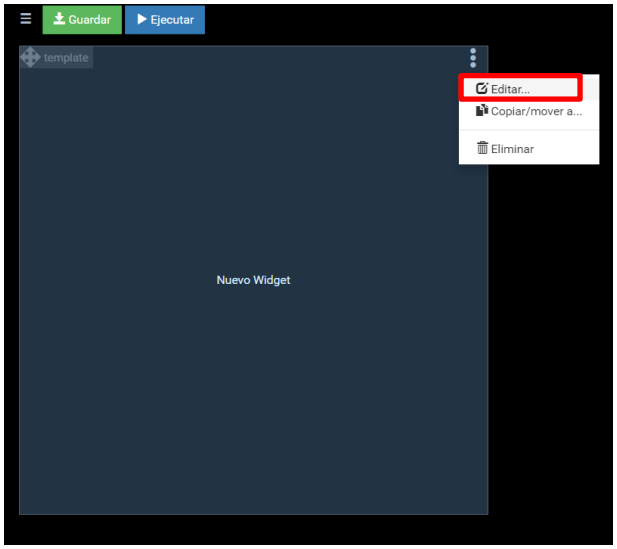

Ilustración 55. Editar widget

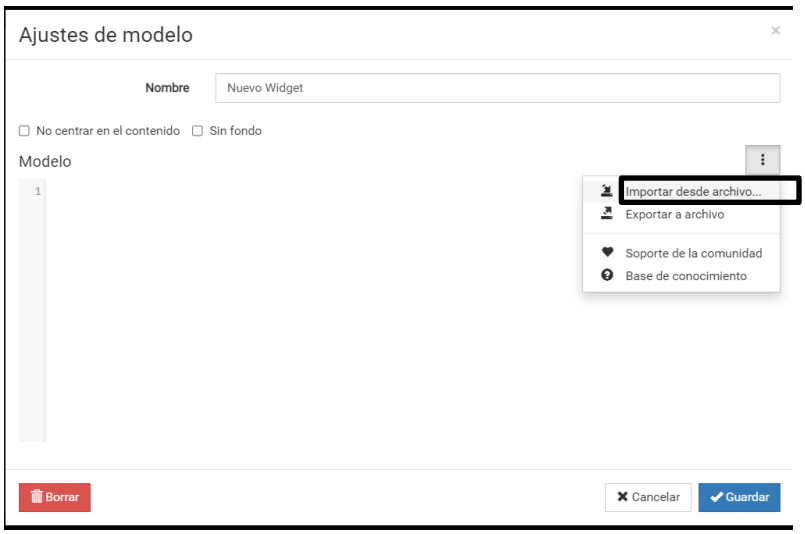

Ilustración 56. Importación de "*3d-view.tpl.html* "

7. Finalmente modificamos la ruta de "lib-url" y "sh3d-url" hacia la carpeta "conf/html" que es donde está el archivo .sh3d de la vivienda a controlar, añadiendo "/static" antes de la ruta donde están los archivos, como aparece en la ilustración 57.

```
<div oc-lazy-load="'/static/sweethome3d.directive.js'">
  <sweet-home-3d
      style="position: absolute; left: 0; right: 0; top: 0; bottom: 0"
      lib-url="/static/sweethome3d/lib"
      sh3d-url="/static/SweetHome3DExample5.sh3d">
  </sweet-home-3d>
</div>
```

Ilustración 57. Modificación de la ruta de los archivos

5.2.2 Asociar un Item a un dispositivo virtual del plano

Gracias a la incorporación de la vista WebGL 3D del visor Sweet Home 3D JS en los paneles de HABpanel creada por Yannick Schaus podemos asociar Items a objetos virtuales del plano en 3D.

Para asociar un Item con un objeto virtual usamos esta plantilla para cada dispositivo:

```
< script  type = " text / ng-template " id = " sweethome3d / My Object " >
        Tu plantilla . . .
</ script >
```

Donde "`My Object`" es el nombre que se le da al objeto virtual en la creación del plano, por lo que hay que abrirlo con Sweet Home 3D y ver el nombre de cada objeto que se quiera usar para controlar un Item.

Dentro del script se especifica el Item correspondiente a ese objeto, la acción a realizar al pulsar sobre él o como se quiere que muestre la información de su estado. Todo esto dependerá del Item seleccionado y lo que queremos obtener de él. A continuación, se muestra en la ilustración 58 un ejemplo de un Item de tipo Switch asociado a una lámpara del plano en 3D.

```
<!--Lamparitas de las mesitas de noche On/Off-->
<script type="text/ng-template" id="sweethome3d/Lamp">
  <div style="min-height: 100px" ng-init="onoff={name: 'Lampara Salon', item: 'LoxoneMiniserver_SalonLamparaSalon',
   hidelabel: 'false', hideonoff: 'false', iconset:'eclipse-smarthome-classic',icon: 'light', icon_size: 75 }">
    <div ng-style="{'background-color': (itemState('LoxoneMiniserver_SalonLamparaSalon')=='ON') ? 'LightGoldenRodYellow' : 'DarkSlateGray' }" style="width:
    200px; height: 75px;">
     <div class="value" style="font-size: 32pt;">{{(itemState('LoxoneMiniserver_SalonLamparaSalon',true) == 'ON') ? 'Encendido' : 'Apagado'}}
      <div class="box switch" style="width: 200px; height: 100px;">
       <div class="switch-content">
        <widget-switch ng-model="onoff"></widget-switch>
       </div>
      </div>
     </div>
    </div>
  </div>
</script>
```

Ilustración 58. Ejemplo de un Item asociado a una lampara

Para interactuar con este elemento hay que pulsar sobre alguna de las lamparitas de las mesitas de noche en el dormitorio y posteriormente pulsar sobre la bombilla que aparece en pantalla y se podrá encender o apagar. Esta visualización se representa en la ilustración 59.

Ilustración 59. Visualización del control de la luminaria

5.3 IU Basic

IU Basic [6] es una interfaz web, basada en Material Design Lite de Google, que permite organizar de forma gráfica los ítems creados, de forma ordenada y siguiendo el orden establecido en un Sitemap diseñado por nosotros.

Su interfaz está basada en AJAX y optimizada para dispositivos con pantallas de todos los tamaños. No permite cambios directos en la configuración de OpenHAB por lo que resulta muy útil para evitar que el usuario final modifique sin querer algún aspecto de la configuración con resultados catastróficos.

Realiza de forma automática la actualización en vivo del estado de los ítems.

Se puede configurar un Sitemap por defecto o podemos seleccionar de una lista el Sitemap que queramos abrir cada vez que accedamos a IU Basic.

5.4 Creación de la interfaz en IU Basic

Para la creación de la interfaz de UI Basic se siguen estos pasos:

1. Seleccionar “Pages” en el menú de la izquierda.
2. Pulsar en el botón azul de la esquina derecha y seleccionar “Create sitemap” para crear la base donde situar todas las partes de la vivienda.
3. En la nueva página pulsar sobre “New Sitemap” y aparece la opción de cambiar su ID y su “Label”, que será su nombre. Justo debajo está la opción para añadir Widget (representado en la ilustración 60), descritos en el apartado 5.1.1.

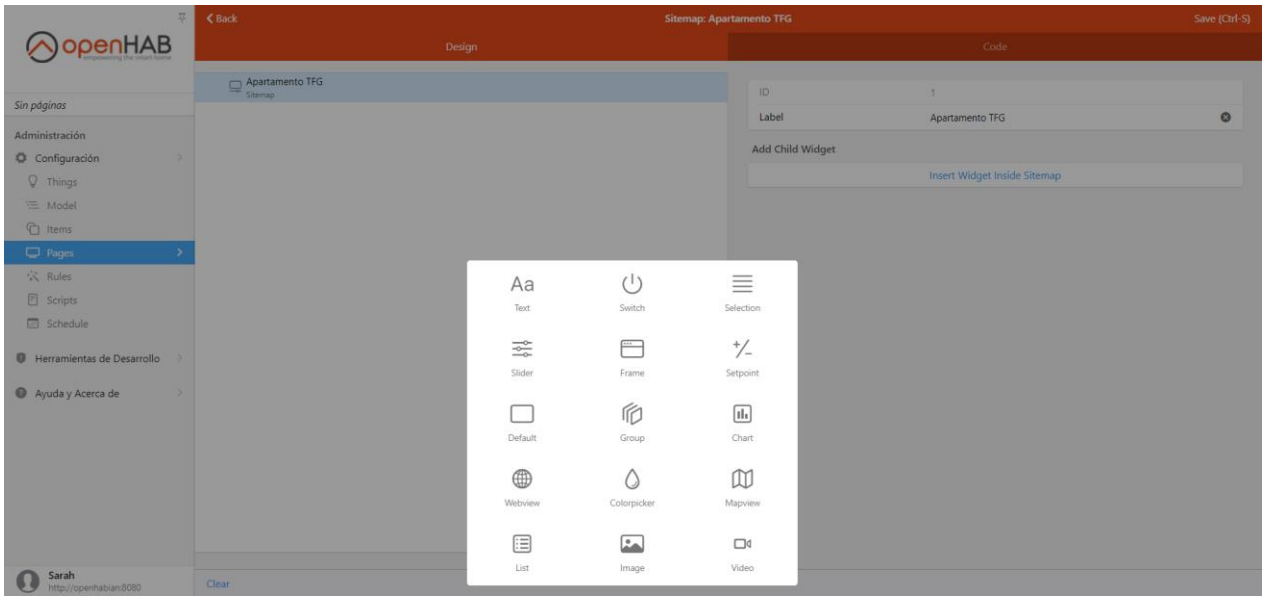

Ilustración 60. Selección del tipo de widget

4. Primero se añade un “Frame” para cada estancia o apartado que queremos tener separado en la interfaz, pulsando siempre en el Sitemap antes de añadir otro “Frame” para evitar crear un “Frame” dentro del creado anteriormente.

5. Dentro de cada “Frame” se selecciona el icono para representar dicho “Frame” y se insertan los widgets que corresponden a cada estancia. A continuación, se describen algunos de los widgets creados:

a. *Setpoint*: Para el control de las persianas del Salón. Label será "Persianas" y se selecciona el Item ya creado para el control de las persianas, se le asigna el icono de las persianas y se establece el porcentaje que se quiere que se cierren o se abran al pulsar una vez (en este caso, será el 100 para que se cierren o abran completamente). Está representado en la ilustración 61.

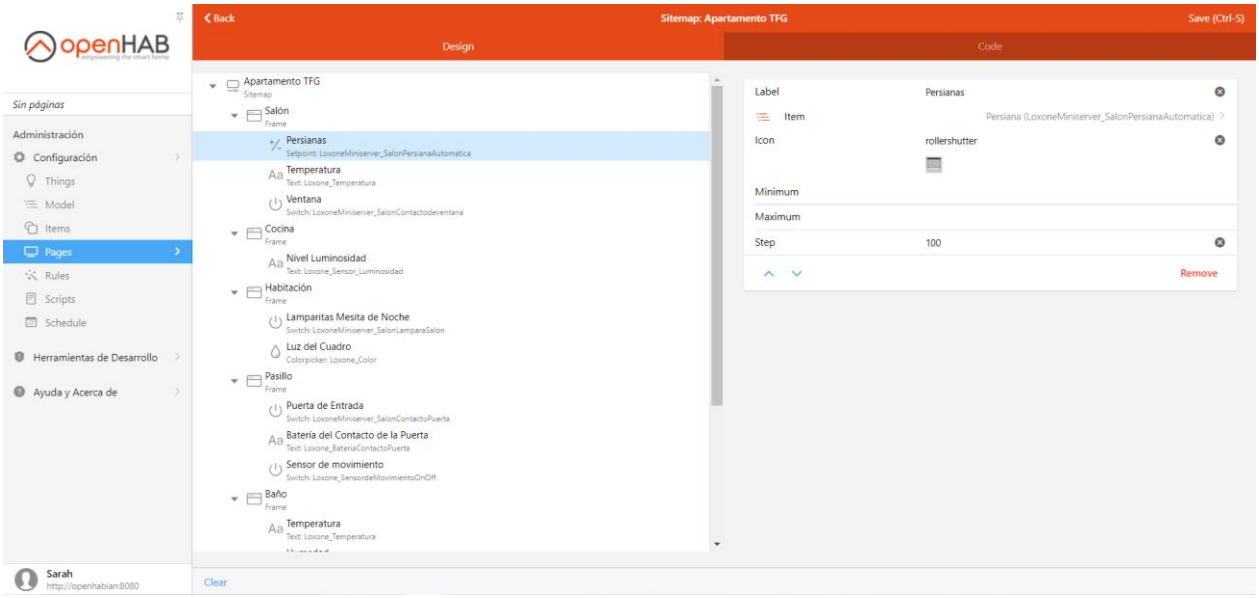

Ilustración 61. Widget para las persianas

b. *Text*: Para la consulta de la temperatura del Salón. Label será "Temperatura" y se selecciona el Item creado para obtener la temperatura, así como un icono característico para representarlo.

c. *Switch*: Para el control del sensor de movimiento del pasillo, representado en la ilustración 62. Label será "Sensor de movimiento" y está asociado al Item del sensor de movimiento para poder decidir si se desea que realice la acción asociada a su activación, normalmente asociado al encendido de la luminaria.

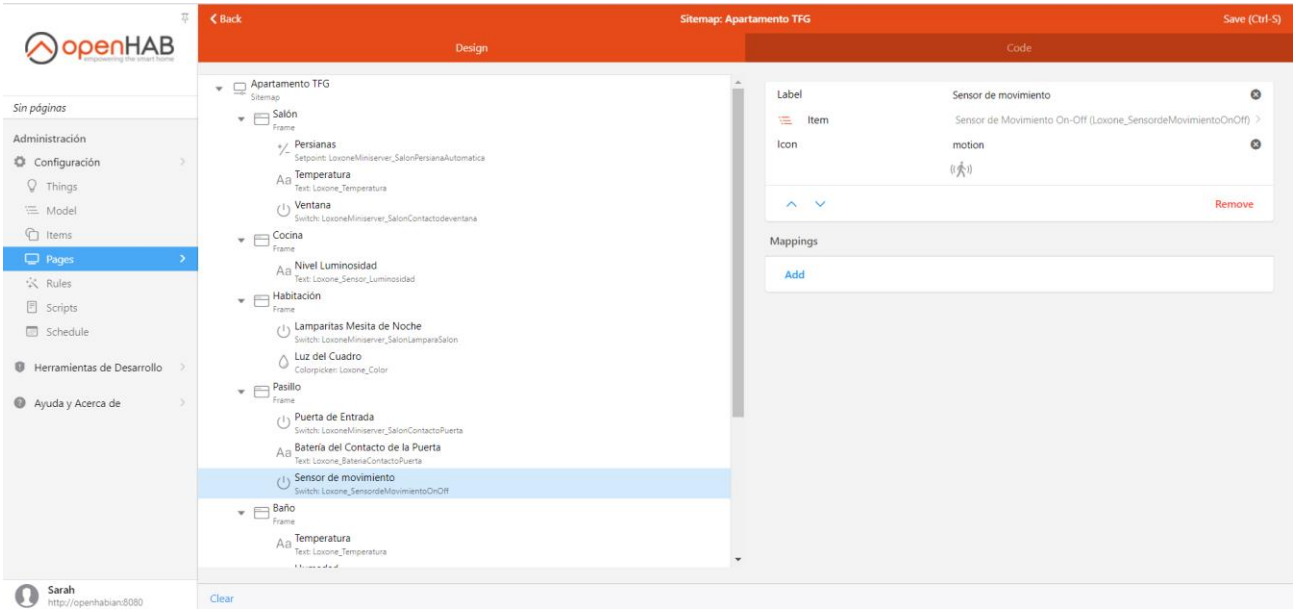

Ilustración 62. Widget para el sensor de movimiento

6. Finalmente, el resultado obtenido usando los dispositivos Loxone, Z-wave y Servidores web (para datos del clima exterior, la fecha y hora actual) se puede ver en la ilustración 63.

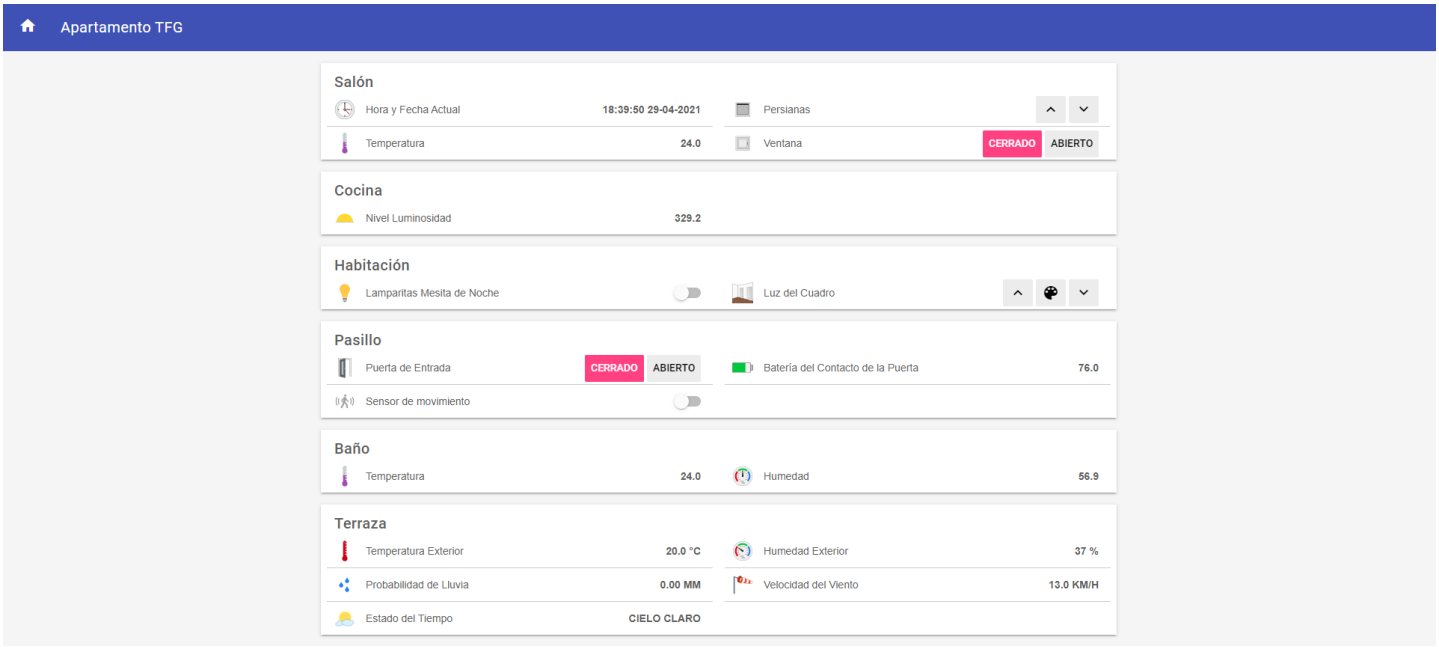

Ilustración 63. Interfaz UI Basic

6 CONCLUSIONES Y LÍNEAS DE FUTURO

Tras la realización de este proyecto queda demostrada la posibilidad de poder controlar el sistema de automatización del hogar Loxone y el sistema Z-wave bajo un mismo sistema ajeno a estos, dando solución al problema de tener que elegir entre uno de estos dos sistemas y no poder integrar dispositivos ajenos a dicho sistema. Gracias a OpenHAB podemos decidir crear nuestro propio sistema de control domótico de la vivienda usando cualquier tipo de sistema domótico, dispositivo o incluso Servicios Web que sean compatibles con OpenHAB y a la vez configurar fácilmente la interfaz a nuestro gusto, pudiendo usar un plano 3D de nuestra vivienda si quisiéramos.

Como línea de futuro sería interesante comprobar la compatibilidad de openHAB 3 con dispositivos que usen el protocolo de comunicación Zigbee, por su bajo consumo, y especialmente su compatibilidad con los asistentes virtuales controlados por voz como Google Home o Alexa, ya que la sociedad actualmente está aceptando estos dispositivos de manera muy favorable para el control de sus viviendas.

REFERENCIAS

[1] https://www.openhab.org/

[2] https://www.loxone.com/

[3] J.M. Maestre, Domótica para Ingenieros, capítulo 6 "Z-wave"

[4] https://www.openhab.org/docs/ui/habpanel/habpanel.html

[5] https://ubunlog.com/sweet-home-3d-interiores-ubuntu/

[6] https://master--openhab-docs-preview.netlify.app/docs/tutorial/uis.html

[7] https://www.openhab.org/docs/installation/openhabian.html

[8] https://sourceforge.net/projects/sweethome3d/files/SweetHome3D-viewer/SweetHome3DJSViewer-5.5.zip/download?use_mirror=netactuate

[9] https://github.com/ghys/habpanel-3dview/blob/master/sweethome3d.directive.js

[10] https://github.com/ghys/habpanel-3dview/blob/master/3d-view.tpl.html

ANEXO

Código usado para la asociación de los Items con los diferentes objetos virtuales del plano en 3D:

```
<style>
    .sweethome3d-modal .modal-dialog {
        width: 200px;
 }
    .sweethome3d-modal .modal-content {
        background-color: black;
 }
</style>


<!--Estado del Sensor de Agua-->
<script type="text/ng-template" id="sweethome3d/Container">
    <div>
        <div class="box dummy" style="width: 200px; height: 200px;">
            <div class="dummy-content" style="font-size: 25px">
                <dt> Sensor de agua </dt>
                <widget-icon iconset="'eclipse-smarthome-classic'" icon="'water'" size="60" center="true">
                </widget-icon>
                <div class="value" style="font-size: 36pt">
                {{(itemState('ZWaveFloodAlarm',true) == 'ON') ? 'Alarma' : 'Ok'}}
                </div>
            </div>
        </div>
    </div>
</script>
```

```
<!--El reloj nos da la fecha actual y la hora-->
<script type="text/ng-template" id="sweethome3d/Clock">
    <div style="min-height: 100px;" ng-init="Reloj={name: 'Hora y Fecha Actual', item: 'HoraLocal_Date',
  'subTextEnabled': true, 'readOnly': true, iconset:'eclipse-smarthome-classic',icon: 'time', icon_size: 75 }">
        <div class="box" style="width: 350px; height: 200px;">
            <div>
                <div>
                    <widget-dummy ng-model="Reloj" style="font-size: 22pt;"></widget-dummy>
            </div>
        </div>
    </div>
</script>
```

```
<!--Lampara del cuadro en Habitación para elegir el color de la misma así como la intensidad-->
<script type="text/ng-template" id="sweethome3d/Wall lamp artwork">
    <div ng-init="piker={item: 'Loxone_Color', style:'aCKolor'}">
        <div class="box-dummy" style="width: 200px; height: 150px; ">
            <div class="dummy-content" style ="font-size:25px; background: DarkSlateGray; color:
            LightSkyBlue;" >
            <dt > Lampara de Colores </dt>
            <widget-colorpicker style= "position:absolute; left: 60px; top: 70px;" ng-model="piker"></widget-
            colorpicker >
            </div>
        </div>
    </div>
</script>
```

```
<!--Lampara del techo Salón y Baño, elección de Escenas-->
<script type="text/ng-template" id="sweethome3d/Pendant lamp">
    <div class="box dummy" style="width: 445.5px; height: 50px; background: DarkSlateBlue;">
        <div class="dummy-content" style="font-size: 30px">
            <dt> Ambientes</dt>
            <div class="box dummy" style="width: 445.4px; height: 40px;">
                <div class="btn-group btn-group-lg">
                    <button type="button" class="btn btn-primary btn-lg" ng-click="sendCmd
                    ('Loxone_ControlIluminacion', '777')">Encendido</button>
                    <button type="button" class="btn btn-primary btn-lg" ng-click="sendCmd
                    ('Loxone_ControlIluminacion', '1')">General</button>
                    <button type="button" class="btn btn-primary btn-lg" ng-click="sendCmd
                    ('Loxone_ControlIluminacion', '2')">Creativa</button>
                     <button type="button" class="btn btn-primary btn-lg" ng-click="sendCmd
                    ('Loxone_ControlIluminacion', '3')">Relax</button>
                    <button type="button" class="btn btn-primary btn-lg" ng-click="sendCmd
                    ('Loxone_ControlIluminacion', '778')">Off</button>
                    <div style="width: 445.5px; height: 150px; background: DarkSlateBlue;" ng-init="luz=
                    {name:'Nivel de Luminosidad', item: 'ZWaveSensorluminance'}">
                            <widget-dummy ng-model="luz" style="font-size: 15pt; position:absolute; left: 110px;
                            top: 60px;"></widget-dummy>
                      </div>
                </div>
            </div>
        </div>
    </div>
</script>
```

```
<!--Mover cortinas-->
<script type="text/ng-template" id="sweethome3d/Curtain">
    <div class="box dummy" style="width: 300px; height: 200px;">
        <div class="dummy-content" style="font-size: 25pt; "> Persiana
            <div class="shutterWidget"><span>
                <button style="size: 20; background-color: LightSkyBlue" ng-click="sendCmd
                ('LoxoneMiniserver_SalonPersianaAutomatica', 'UP')" >
                    <i class="glyphicon glyphicon-menu-up"></i></button>
                <button ng-click="sendCmd('LoxoneMiniserver_SalonPersianaAutomatica', 'STOP')">
                    <widget-icon iconset="'eclipse-smarthome-classic'" icon="'blinds'" size="125" center="true"
                    state="itemValue('LoxoneMiniserver_SalonPersianaAutomatica')" /></button>
                <button style="margin-right: 0; size: 20; background-color: LightSkyBlue" ng-click="sendCmd
                ('LoxoneMiniserver_SalonPersianaAutomatica', 'DOWN')">
                    <i class="glyphicon glyphicon-menu-down"></i></button>
                </span>
            </div>
        </div>
    </div>
</script>


<!--Radiadores Eléctricos marca la temperatura actual (Z-wave) -->
<script type="text/ng-template" id="sweethome3d/Electric radiator">
    <div style="min-height: 200px" ng-init="model={name: 'Temperatura', item: 'ZWave_temperature',
     unit:'°', 'subTextEnabled': true, 'step': 0.5, 'min': 12, 'max': 30, 'readOnly': true}">
        <div class="box" style="width: 300px; height: 300px;">
            <div>
                <div>
                    <widget-knob ng-model="model"></widget-knob>
                </div>
            </div>
        </div>
    </div>
</script>
```

```
<!--Ventana Exterior, temperatura, humedad, velocidad del viento, probabilidad de lluvia y estado del tiempo exterior-->
<script type="text/ng-template" id="sweethome3d/Fixed window">
    <div style="min-height: 50px; position:absolute; left: 20px; top: 20px;" ng-init="Temp={name: 'Temperatura Ext', item: 'OneCallAPIweatherandforecast_OutdoorTemperature', 'subTextEnabled': true, 'readOnly': true, iconset:'eclipse-smarthome-classic',icon: 'temperature', icon_size: 75 }">
        <div style="min-height: 50px; position:absolute; left: 20px; top: 20px;" ng-init="Hum={name: 'Humedad Ext', item: 'OneCallAPIweatherandforecast_AtmosphericHumidity', 'subTextEnabled': true, 'readOnly': true, iconset:'eclipse-smarthome-classic',icon: 'humidity', icon_size: 75 }">
            <div style="min-height: 50px; position:absolute; left: 20px; top: 120px;" ng-init="Lluvia={name: 'Probabilidad de lluvia', item: 'OneCallAPIweatherandforecast_Rain', 'subTextEnabled': true, 'readOnly': true, iconset:'eclipse-smarthome-classic', icon: 'rain', icon_size: 75 }">
                <div style="min-height: 50px; position:absolute; left: 200px; top: 120px;" ng-init="Viento={name: 'Velocidad del Viento', item: 'OneCallAPIweatherandforecast_WindSpeed', 'subTextEnabled': true, 'readOnly': true, iconset:'eclipse-smarthome-classic',icon: 'wind', icon_size: 75 }">
                <div style="min-height: 50px; position:absolute; left: 100px; top: 220px;" ng-init="Estado={name: 'Estado del tiempo', item: 'OneCallAPIweatherandforecast_WeatherCondition', 'subTextEnabled': true, 'readOnly': true, iconset:'eclipse-smarthome-classic',icon: 'sun_clouds', icon_size: 75  }">
                    <div class="box" style="width: 600px; height: 720px; position:absolute; right: 100px; bottom: 1px;">
                        <div>
                            <div>
                                <widget-dummy ng-model="Temp" style="font-size: 22pt; position:absolute; left: 50px; top: 155px;  "></widget-dummy>
                                <widget-dummy ng-model="Hum" style="font-size: 22pt; position: absolute; left: 350px; top: 155px; "></widget-dummy>
                                <widget-dummy ng-model="Lluvia" style="font-size: 22pt; position: absolute; left: 20px; top: 320px; "></widget-dummy>
                                <widget-dummy ng-model="Estado" style="font-size: 22pt; position: absolute; left: 330px; top: 320px;"></widget-dummy>
                                <widget-dummy ng-model="Viento" style="font-size: 22pt; position: absolute; left: 10px; top: 500px; "></widget-dummy>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
</script>
```

```
<!--Radiador del Baño marca la temperatura actual y la humedad-->
<script type="text/ng-template" id="sweethome3d/Radiator">
    <div style="min-height: 100px; position:absolute; left: 20px; top: 20px;" ng-init="Temp={name:
    'Temperatura', item: 'Loxone_Temperatura', 'subTextEnabled': true, 'readOnly': true, iconset:'eclipse-
    smarthome-classic',icon: 'temperature', icon_size: 75 }">
        <div style="min-height: 100px; position:absolute; left: 200px; top: 20px;" ng-init="Hum={name:
        'Humedad', item: 'Loxone_Humedad', unit:'%', 'subTextEnabled': true, 'readOnly': true,iconset:'eclipse-
        smarthome-classic',icon: 'humidity', icon_size: 75  }">
            <div class="box" style="width: 400px; height: 220px;">
                <div>
                    <div>
                        <widget-dummy ng-model="Temp" style="font-size: 22pt; position:absolute; left: 20px;
                        top: 20px;  "></widget-dummy>
                            <widget-dummy ng-model="Hum" style="font-size: 22pt; position: absolute; left:
                            220px; top: 20px; "></widget-dummy>
                    </div>
                </div>
            </div>
        </div>
    </div>
</script>


<!--Pulsador Baño para activar o desactivar el sensor de movimiento-->
<script type="text/ng-template" id="sweethome3d/Light switch">
    <div style="min-height: 75px" ng-init="onoff={name: 'On/Off Sensor Movimiento', item:
    'Loxone_SensordeMovimientoOnOff', hidelabel: 'true', hideonoff: 'true', iconset:'eclipse-smarthome-
    classic',icon: 'motion', icon_size: 75 }">
        <div ng-style="{'background-color': (itemState('Loxone_SensordeMovimientoOnOff')=='ON') ?
        'Cornsilk' : 'DarkSlateGray' } " style="width: 300px; height: 100px;">
            <div class="value" style="font-size: 32pt; width: 300px; height: 100px;">
            {{(itemState('Loxone_SensordeMovimientoOnOff',true) == 'ON') ? 'Activado' : 'Desactivado'}}
                <div class="box switch" style="width: 300px; height: 100px;">
                    <div class="switch-content">
                            <widget-switch ng-model="onoff"></widget-switch>
                    </div>
            </div>
        </div>
    </div>
</div>
</script>
```

```
<!--Lamparitas de las mesitas de noche On/Off-->
<script type="text/ng-template" id="sweethome3d/Lamp">
    <div style="min-height: 100px" ng-init="onoff={name: 'Lampara Salon', item:
    'LoxoneMiniserver_SalonLamparaSalon',  hidelabel: 'false', hideonoff: 'false', iconset:'eclipse-smarthome-
    classic',icon: 'light', icon_size: 75 }">
        <div ng-style="{'background-color': (itemState('LoxoneMiniserver_SalonLamparaSalon')=='ON') ?
        'LightGoldenRodYellow' : 'DarkSlateGray' }" style="width: 200px; height: 75px;">
            <div class="value" style="font-size: 32pt;">
            {{(itemState('LoxoneMiniserver_SalonLamparaSalon',true) == 'ON') ? 'Encendido' : 'Apagado'}}
                <div class="box switch" style="width: 200px; height: 100px;">
                    <div class="switch-content">
                        <widget-switch ng-model="onoff"></widget-switch>
                    </div>
                </div>
            </div>
        </div>
    </div>
</script>


<!--Estado de las Ventanas-->
<script type="text/ng-template" id="sweethome3d/Slider window">
    <div>
        <div class="box dummy" style="width: 300px; height: 300px;">
            <div class="dummy-content">
                <dt> Ventana del Salón </dt>
                <widget-icon iconset="'eclipse-smarthome-classic'" icon="'window'" size="200" center="true"
                state="(itemState('LoxoneMiniserver_SalonContactodeventana') == 'OFF') ? 'OPEN' :
                'CLOSED'"></widget-icon>
                <div class="value" style="font-size: 36pt">
                    {{(itemState('LoxoneMiniserver_SalonContactodeventana', true) == 'OFF') ? 'Abierta' :
                    'Cerrada'}}
                    </div>
                </div>
            </div>
        </div>
</script>
```

```
<!--Estado de la Puerta Exterior-->
<script type="text/ng-template" id="sweethome3d/Exterior door">
    <div>
        <div class="box dummy" style="width: 300px; height: 300px;">
            <div class="dummy-content">
                Entry Door
                <widget-icon iconset="'eclipse-smarthome-classic'" icon="'door'" size="200" state=
                "(itemState('LoxoneMiniserver_SalonContactoPuerta') == 'OFF') ? 'OPEN' :
                'CLOSED'"></widget-icon>
                <div class="value" style="font-size: 36pt">
                {{(itemState('LoxoneMiniserver_SalonContactoPuerta',true) == 'OFF') ? 'Abierta' : 'Cerrada'}}
                </div>
            </div>
        </div>
    </div>
</script>

<!--Cambio de la intensidad de la luz de la pared-->
<script type="text/ng-template" id="sweethome3d/Wall uplight">
    <div style="min-height: 200px" ng-init="model={name: 'Wall light', item:
    'LoxoneMiniserver_SalonLamparaSalon', vertical: true, step: 10, showticks: true}">
        <div class="box" style="width: 300px; height: 300px;">
            <widget-slider ng-model="model"></widget-slider>
        </div>
    </div>
</script>



<div oc-lazy-load="'/static/sweethome3d.directive.js'">
    <sweet-home-3d
        style="position: absolute; left: 0; right: 0; top: 0; bottom: 0"
        lib-url="/static/sweethome3d/lib"
        sh3d-url="/static/SweetHome3DExample5.sh3d">
    </sweet-home-3d>
</div>
```