OPEN SOURCE SMART HOME SYSTEM (Arduino - Raspberry Pi - OpenHAB)

José Fernando Adrán Otero
Grado de Tecnologías de las Telecomunicaciones
Arduino

Antoni Morell Pérez
Pere Tuset Peiró

10/06/2018

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	*OPEN SOURCE SMART HOME SYSTEM (Arduino - Raspberry Pi - OpenHAB)*
Nombre del autor:	*José Fernando Adrán Otero*
Nombre del consultor/a:	*Antoni Morell Pérez*
Nombre del PRA:	*Pere Tuset Peiró*
Fecha de entrega (mm/aaaa):	06/2018
Titulación::	*Grado de Tecnologías de las Telecomunicaciones*
Área del Trabajo Final:	*Arduino*
Idioma del trabajo:	*Español*
Palabras clave	*Domótica, OpenHab*

Resumen del Trabajo:

El objetivo principal de este trabajo es el de implementar un sistema domótico en un hogar, de una forma práctica y económica, haciendo uso de las nuevas tecnologías Opensource, las cuales se adaptan perfectamente a las necesidades de las Smart Home.

Lo primero que se verá serán los elementos que compondrán el sistema domótico, tanto placas como sensores y actuadores. Se utilizarán 2 tipos de placas controladoras: Arduino – para recibir información de los sensores, y enviar órdenes a los actuadores – y Raspberry Pi, la cual tendrá la responsabilidad de consolidar toda la información que reciba de los Arduino y tomará las decisiones según los parámetros que haya configurados.

Para realizar la comunicación entre ambos dispositivos se utilizará la plataforma Opensource “OpenHAB”, ya que es directamente compatible con el cerebro del sistema, la Raspberry Pi, y además dispone de múltiples interfaces (Bindings) para controlar diferentes dispositivos con conexión a la red, lo que supone un paso más en la domotización del hogar.

Para realizar la comunicación entre los diferentes dispositivos, Arduino y Raspberry, se hará uso del protocolo MQTT. Como se considera la vivienda de obra nueva, toda la red será cableada.

Una vez establecida la comunicación, se realizará la integración de los diferentes dispositivos y su configuración.

Abstract:

The main objective of this work is to implement a Home Automation system in a home, in a practical and economical way, using Opensource technologies, which adapt perfectly to the requests of the Smart Home.

The first thing we are going to see will be the elements that will structure the home automation system, like boards, sensors and actuators. We will use 2 kind of controller boards: Arduino - to receive information from the sensors, and send orders to the actuators - and Raspberry Pi, which will have the responsibility of merging all the information it receives from the Arduino and it will make the decisions according to the parameters that we will configure

To communicate both devices we will use the Opensource platform "OpenHAB", because it is directly compatible with the brain of our system, the Raspberry Pi, and also has multiple interfaces (Bindings) to control diverse devices with connection to the network, what supposes an advance in the automation of the home.

To make the communication between the different controller boards, Arduino and Raspberry, we will use the MQTT protocol. As we consider new construction housing, and the entire network will be wired.

Once the communication is established, it will begin with the integration of the different devices and their configuration.

INDICE DE CONTENIDO

FICHA DEL TRABAJO FINAL i
Resumen del Trabajo: i
Abstract: ii
INDICE DE CONTENIDO 1
INDICE DE FIGURAS 3
INDICE DE TABLAS 3
1.1 Descripción del Proyecto: 4
1.2 Propuesta de Trabajo: 4
1.3 Objetivos del proyecto 7
1.3.1 Requisitos: 8
1.4 Viabilidad: 9
1.5 Producto Obtenido 10
1.6 Descripción de la memoria 11
1.7 Planificación 12
CAPÍTULO 2 – ESTADO DEL ARTE 14
2.1 Domótica 14
2.2 Sistemas o protocolos dominantes 15
2.3 Arquitecturas implementables 16
CAPÍTULO 3 - DISEÑO 17
3.1 Diseño del sistema 17
3.1.1 Diseño de la arquitectura 18
3.1.2 Configuración de Red 19
3.1.3 Diseño de la maqueta 20
3.1.3 Funcionamiento básico del sistema 21
3.2 Tecnologías y componentes utilizados 22
3.2.1 Arduino 22
3.2.2 Sensores 23
3.2.2.1 Temperatura y humedad 23
3.2.2.2 Luminosidad 24
3.2.2.3 PIR, o sensor de movimiento 24
3.2.2.4 Sensor de llama 24
3.2.2.5 Sensor de Gas 25
3.2.2.6 Sensor de Inundación 25
3.2.2.7 Otros sensores 25
3.2.3 Actuadores 26
3.2.3.1 Relés 26
3.2.3.2 Buzzer 26
3.2.3.3 Diodos led 27

3.2.3.4 Otros actuadores ... 27
3.2.4 Raspberry Pi ... 27
3.2.5 OpenHAB ... 28
3.2.6 Protocolo MQTT ... 29
CAPÍTULO 4 - IMPLEMENTACIÓN ... 30
4.1 Interconexión entre los nodos principales ... 30
4.1.1 Pruebas de conexión OpenHAB—MQTT ... 30
4.1.2 Configuración adicional Openhab ... 32
4.1.3 Programación de entorno OpenHAB ... 32
4.1.3.1 Configuración ... 32
4.1.3.2 Interfaz ... 34
4.2 Inicialización y configuración de sensores y actuadores en Arduino ... 36
4.2.1 Arduino – Código genérico ... 37
4.2.2 Uso de Ethernet Shield con Arduino Mega ... 38
4.2.3 Código Arduino para MQTT full dúplex ... 38
4.2.4 Uso de protocolo SPI ... 39
4.2.5 Código Habitación Principal: ... 41
4.2.6 Código Simulación ... 43
4.3 Pruebas E/S ... 45
4.4 Configuración de reglas ... 46
4.5 Implementación de gráficas en Grafana ... 47
4.6 Streaming de Video IP ... 49
4.7 Creación de la Maqueta ... 50
4.5.1 Diseño maqueta planta baja ... 50
4.5.2 Diseño maqueta habitación principal ... 51
CAPÍTULO 5 – PRUEBAS Y RESULTADOS ... 54
5.1 Pruebas del sistema ... 54
5.2 Resultados ... 57
BIBLIOGRAFÍA ... 58
ANEXO I – Instalación Básica ... 60
ANEXO II – Instalación OpenHab ... 64
ANEXO III – Instalación bróker MQTT ... 68
ANEXO IV – Instalación Samba ... 69
ANEXO V - Configuración de Openhab para conectividad ... 71
ANEXO VI – Servicios de persistencia (rrd4j, mapDB, InfluxDB) ... 73
ANEXO VII – Grafana ... 76

INDICE DE FIGURAS

Figura 1 - Esquema Solución básica 6
Figura 2 - Esquema ejemplo solución 6
Figura 3 - Arquitecturas Domóticas 16
Figura 4 - Esquema Equipamiento de control 18
Figura 5 - Capas OSI del sistema 18
Figura 6 - Configuración de Red 19
Figura 7 - Maqueta SketchUp 20
Figura 8 - Capas OSI 21
Figura 9 - Esquema Completo de la solución 21
Figura 10 - Arduino Mega 22
Figura 11 - W5100 23
Figura 12 - DHT11, DHT22, y conexiones DHT22 23
Figura 13 - LDR55 24
Figura 14 - PIR 24
Figura 15 - Funcionamiento PIR 24
Figura 16 - Sensor de Llama 24
Figura 17 - Sensor de Gas MQ135 25
Figura 18 - Sensor de Inundación 25
Figura 19 - Relé de 5V 26
Figura 20 - Funcionamiento Buzzer 26
Figura 21 - Diodo Led 27
Figura 22 - Raspberry Pi 3 B 27
Figura 23 - Pantalla OpenHab 28
Figura 24 - Esquema MQTT del sistema 29
Figura 25 - Prueba conectividad MQTT 31
Figura 26 - OpenHab - Basic UI 35
Figura 27 - OpenHab - HabAdmin 36
Figura 28 - Programación Arduino 37
Figura 29 - Bus SPI 40
Figura 30 - Pruebas Sistema 45
Figura 31 - Gráficas Sistema 48
Figura 32 - Video Camara IP 49
Figura 33 - Maqueta 50
Figura 34 - Esquema Electrico Luz 51
Figura 35 - Esquema Prototipo 53
Figura 36 - ESP8266 57

INDICE DE TABLAS

Tabla 1 – Requisitos de la solución 8
Tabla 2 - Diferencias entre protocolos actuales 15
Tabla 3 - Tipos de Arquitecturas implementables 16
Tabla 4 - Configuración de Red 19
Tabla 5 - Arduino Mega 22
Tabla 6 - Resultados 57

CAPÍTULO 1 – INTRODUCCIÓN

1.1 Descripción del Proyecto:

El objetivo principal de este trabajo es el de implementar un sistema domótico en un hogar, de una forma práctica y económica, haciendo uso de las nuevas tecnologías Opensource, las cuales se adaptan perfectamente a las necesidades de las Smart Home.

Se usarán 2 tipos de placas controladoras: Arduino – para recibir información de los sensores, y enviar órdenes a los actuadores – y Raspberry Pi, la cual tendrá la responsabilidad de consolidar toda la información que reciba de los Arduino y tomará las decisiones según los parámetros que se configuren.

Para realizar la comunicación entre ambos dispositivos se utilizará la plataforma Opensource "OpenHAB", ya que es directamente compatible con el cerebro del sistema, la Raspberry Pi, y además dispone de múltiples interfaces (Bindings) para controlar diferentes dispositivos con conexión a la red.

Para realizar la comunicación entre los diferentes dispositivos, Arduino y Raspberry, se hará uso del protocolo MQTT. Se considera una vivienda de obra nueva, con tubos según normativa para realizar una instalación cableada basada en protocolo TCP-IP.

1.2 Propuesta de Trabajo:

El sistema consistirá en una arquitectura descentralizada, usando una Raspberry Pi como nodo central, ubicada en el cuarto de instalaciones, coincidente con la ubicación del cuadro eléctrico, y un controlador Arduino para cada estancia de la vivienda para gestionar los sensores y actuadores mencionados a continuación:

- Entrada y pasillos
    - Sensores: temperatura, humedad, luminosidad y movimiento
    - Actuadores: Relés para control de luz
- Salón – Comedor
    - Sensores: temperatura, humedad, luminosidad y movimiento
    - Actuadores: Relés para control de luz y calefacción

- Cocina
    - Sensores: temperatura, humedad, luminosidad, gases, fuego e inundación
    - Actuadores: Actuador sonoro por fuego y/o humo e inundación
- Baño Principal
    - Sensores: temperatura, humedad, luminosidad e inundación
    - Actuadores: Actuador sonoro por inundación
- Habitación principal
    - Sensores: temperatura, humedad, luminosidad y movimiento
    - Actuadores: Relés para control de luz y calefacción
- Habitación de invitados
    - Sensores: temperatura, humedad, luminosidad y movimiento
    - Actuadores: Relés para control de luz y calefacción

Se montará una pequeña maqueta para poder ver la utilidad del sistema de forma visual.

Cada Arduino se conectará con la Raspberry Pi a través de la red local del hogar, disponiendo así de una dirección IP dedicada para cada elemento. Además, utilizará un direccionamiento estático para albergar estos dispositivos, para darle independencia parcial de los demás equipos de la red local, evitando así posibles conflictos de IP con por ejemplo un teléfono de un invitado que se conecte a la red del hogar.

La pasarela para la comunicación entre los Arduino y la Raspberry será el protocolo MQTT (Message Queue Telemetry Transport), el cual es un protocolo usado para la comunicación machine-to-machine (M2M), orientado a la comunicación de sensores, debido a que consume muy poco ancho de banda y puede ser utilizado en la mayoría de los dispositivos empotrados con pocos recursos (CPU, RAM, …).

Por último, se utilizará la plataforma Opensource OpenHAB como plataforma principal para el sistema, la cual usa Java como lenguaje de programación para incluir los diferentes elementos en el sistema. Esta plataforma está disponible para cualquier sistema operativo (SO), tanto en PC, como en dispositivo móvil, lo cual ofrece una flexibilidad y características que permiten desarrollarlo perfectamente en este proyecto.

El esquema resumido de alto nivel de la conectividad de la solución sería el siguiente:

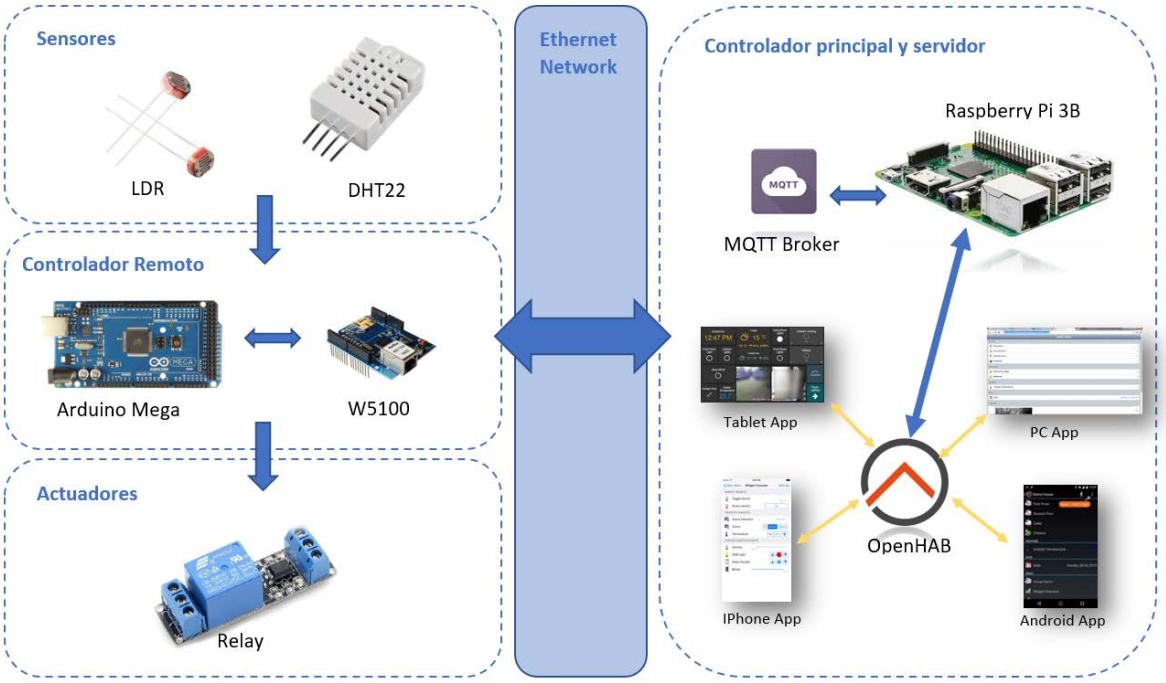


Figura 1 - Esquema Solución básica

Figura 2 - Esquema ejemplo solución

Además, OpenHAB dispone de una arquitectura completamente modular, por lo que es sencillo añadir nuevas funcionalidades al sistema sin necesidad de realizar una reprogramación de los demás elementos, el cual es un punto clave en el desarrollo de este proyecto.

Para la planificación del modelo a ejecutar se diferenciarán varias fases que incluirán:

- Definición de arquitectura de red y diseño a implantar.
- Un estudio de los dispositivos necesarios para la implementación del sistema
- La normativa legal que aplicar
- Presupuesto que incluirá una estimación de costes que reflejarán, de la forma más fiel posible, la viabilidad del proyecto en todo su conjunto
- Por último, la ejecución técnica se realizará en dos fases:
- Se realizará la conexión entre los nodos principales a través del protocolo MQTT.
- Se realizará la programación en los diferentes dispositivos para la adaptación de sensores, actuadores y reglas al sistema.

1.3 Objetivos del proyecto

Uno de los problemas más grandes que tendrá que afrontar Europa en los próximos años está relacionado con el envejecimiento de la población, así como con la dispersión poblacional. Para ello, una de las formas de reducir el impacto en la calidad de vida de las personas puede llegar con la automatización de procesos para aumentar el confort y la seguridad de nuestros hogares. Nuestro proyecto nace de la idea de implementar, mediante proyectos Opensource, una Smart Home de un modo más económico que si se tratase de un sistema tradicional, pero sin perder funcionalidad.

Desde los años 80 se ha venido hablando del concepto “Domótica”, pero no ha llegado a implantarse con fuerza, o por lo menos con toda la que se le presuponía, en parte por los altos costes de instalación y de equipamiento de los que disponen. Protocolos como el KNX o el X10 están ya estandarizados, pero los costes de instalación, y sobre todo de mantenimiento, son muy altos para una gran parte de la población. Para ello, y usando plataformas Opensource, se propone crear un sistema domótico en una vivienda unifamiliar por una décima parte del coste que supondría la instalación de un sistema basado en un protocolo tradicional, con la ventaja además de la libertad de configurar nuestro sistema acorde a nuestras necesidades, de un modo escalable, y sin necesidad de depender de dicho protocolo.

Este TFG se basa en la implementación del sistema en una vivienda unifamiliar, pero se pretende que todo el desarrollo de este sistema se pueda replicar en comercios, oficinas, o locales de otra índole, permitiendo así la universalización de la solución,

aunque los diseñadores particulares deberán adaptarla a sus necesidades específicas. Por ejemplo, dentro de estos grupos, se podría incluir el de implementar este sistema en una residencia de ancianos, un hospital psiquiátrico, centros de internamiento de personas con discapacidades… para realizar la monitorización de los internos, activación o desactivación de luces, enchufes… detectores de presencia… y demás automatizaciones que ayudarían al control de los propios centros de esta índole.

Los objetivos de este proyecto son:

I. Realizar la conexión entre una Raspberry Pi 3 con varios módulos Arduino, a través de un bróker MQTT
II. Realizar la programación de los sensores y actuadores identificados en este proyecto en los Arduino de la vivienda
III. Realizar la comunicación entre todos los elementos del sistema
IV. Establecer reglas en el sistema domótico para activar o desactivar la calefacción automáticamente según la temperatura registrada en cada habitación

1.3.1 Requisitos:

En esta sección se establecen una serie de requisitos medibles que se usarán durante las pruebas del sistema para analizar el logro o no de los resultados esperados:

Nº	Descripción	Resultado
1	Conectividad E2E de la solución usando el protocolo M2M MQTT	
2	Uso de BBDD InfluxDB para servicios de persistencia e interconexión con Grafana	
3	Gestión de servicios de streaming de video IP con la cámara nativa de Raspberry PI	
4	Gestión de encendido y apagado de luces mediante plataforma Openhab, conociendo en cada momento	
5	Activación y desactivación de la alarma del hogar por medio de una tarjeta RFID	
6	Gestión de la calefacción del hogar, de forma personalizada y dependiente de la temperatura de cada estancia	
7	Gestión de alarmas en caso de escape de gases, fuego o inundación	
8	Solución completa con servicios integralmente Open Source	
9	Solución completa con presupuesto <7000€ (Iva incluido)	

Tabla 1 – Requisitos de la solución

1.4 Viabilidad:

El sistema que se desarrollará en este trabajo está basado en la plataforma OpenHAB, que al tratarse de un proyecto Opensource, nos permitirá desarrollar un soporte para nuevos dispositivos, además de proporcionar una aplicación de Android o iOS, una aplicación web, una aplicación de escritorio y un Cloud disponible para acceder a nuestra red desde cualquier punto del planeta. El uso de esta plataforma aporta una interfaz al sistema moderna y amigable, por lo que la implementación de este se convierte en una alternativa real al de un sistema tradicional, de un modo más económico.

Económicamente, los costes del proyecto se pueden diferenciar entre costes de recursos humanos (horas/técnico), y costes del equipamiento necesario para implementar el proyecto:

Costes de los recursos humanos, contando con un precio de 24€/hora de ingeniería:

Nombre de tarea	Tiempo	Coste
Planificación de Proyecto	25 Horas	600,00 €
Diseño del sistema	30 Horas	720,00 €
Interconexión de nodos principales	80 Horas	1.920,00 €
Programación de sensores y actuadores	60 Horas	1.440,00 €
Total	195 Horas	4.680,00 €

Costes de los equipos necesarios para la implementación de proyecto:

Componentes del proyecto	Precio	Ud	Coste
Raspberry Pi 3	34,00 €	1,00	34,00 €
Arduino	20,26 €	6,00	121,56 €
Eth Shield	5,06 €	6,00	30,36 €
Elementos de maqueta	60,00 €	1,00	60,00 €
Sensores y actuadores	82,55 €	1,00	82,55 €
Elementos auxiliares (cables, protoboard, conectores...)	100,00 €	1,00	100,00 €
Total			428,47 €

Por lo que el coste de proyecto ascendería a 5108,47€. Para realizar una estimación grosso modo, se observa que la implementación por habitación supondría un coste aproximado de 836.41€, muy por debajo de un sistema domótico convencional. Buscando un margen cercano al 30% de beneficio, se obtendrá una venta de 1200€ por habitación. Esta estimación es bastante cercana a la realidad, ya que, aunque los costes Hw no dependen directamente del número de estancias, es cierto que aportan una pequeña variación al coste total.

Una vez implementado este sistema, sería sencillo modificarlo para ser usado en un hospital, residencia de ancianos, centro de educación especial… facilitando la labor del personal de dichos centros con la automatización de los sistemas, por lo que el desarrollo de este sistema es viable económica y socialmente.

Por tanto, y teniendo en cuenta los últimos cálculos, sería necesario abordar pocos proyectos para cubrir costes y recuperar la inversión.

Proyectos	Coste	Venta	Margen	Beneficio
Vivienda de 7 estancias	5.959,88 €	8.400,00 €	29%	2.440,12 €
Vivienda de 4 estancias	3.405,65 €	4.800,00 €	29%	1.394,35 €
Residencia de ancianos (100 estancias)	85.141,17 €	120.000,00 €	29%	34.858,83 €
Colegio de educación especial (40 estancias)	34.056,47 €	48.000,00 €	29%	13.943,53 €
Total	128.563,16 €	181.200,00 €	29%	52.636,84 €

En estos costes no se ha contemplado la necesidad de proveer de terminales móviles ni de ordenadores al cliente, puesto que actualmente no se considera una necesidad no cubierta. De hecho, el probable éxito del sistema de basa en la disponibilidad de estos dispositivos por la práctica totalidad de la población que pueda plantearse la implementación de un sistema domótico en su hogar.

1.5 Producto Obtenido

Como salida de este proyecto se obtendrán los siguientes entregables:

- Una memoria técnica que incluye la descripción del sistema, así como su desarrollo y las pruebas realizadas
- Una presentación de recopila los puntos más importantes de este proyecto, además de ejemplos prácticos de la solución
- El propio sistema domótico, a través de su parte lógica, que serán las guías de configuración, código…

1.6 Descripción de la memoria

En el Capítulo 2 se realizará una síntesis del estado actual de las diferentes tecnologías y protocolos que se usan en la actualidad para los proyectos domóticos, así como las topologías de posible implementación en proyectos de esta índole.

En el Capítulo 3 se realizará el diseño de la solución del sistema domótico que será implementada en el TFG.

En este capítulo se realizará una descripción de alto nivel del desarrollo del proyecto, indicando como se estructurará el sistema a nivel físico y de configuración. Además, se expondrán los dispositivos a utilizar, como las propias placas controladoras, la plataforma OpenHAB, y los sensores y actuadores, y el protocolo de comunicación MQTT.

En el Capítulo 4 se abordará el desarrollo del sistema completo, partiendo del sistema Openhab instalado en el nodo principal del sistema (explicado en los anexos), y que constará de 2 partes. Por un lado, la configuración del sistema en Raspberry Pi –Openhab y MQTT- y por otro la programación y configuración de los nodos remotos sobre plataforma Arduino.

Por último, en este mismo capítulo, se expondrán unas pruebas básicas de conectividad E2E, así como la configuración de reglas sobre el sistema completo y la base de creación de la maqueta del sistema.

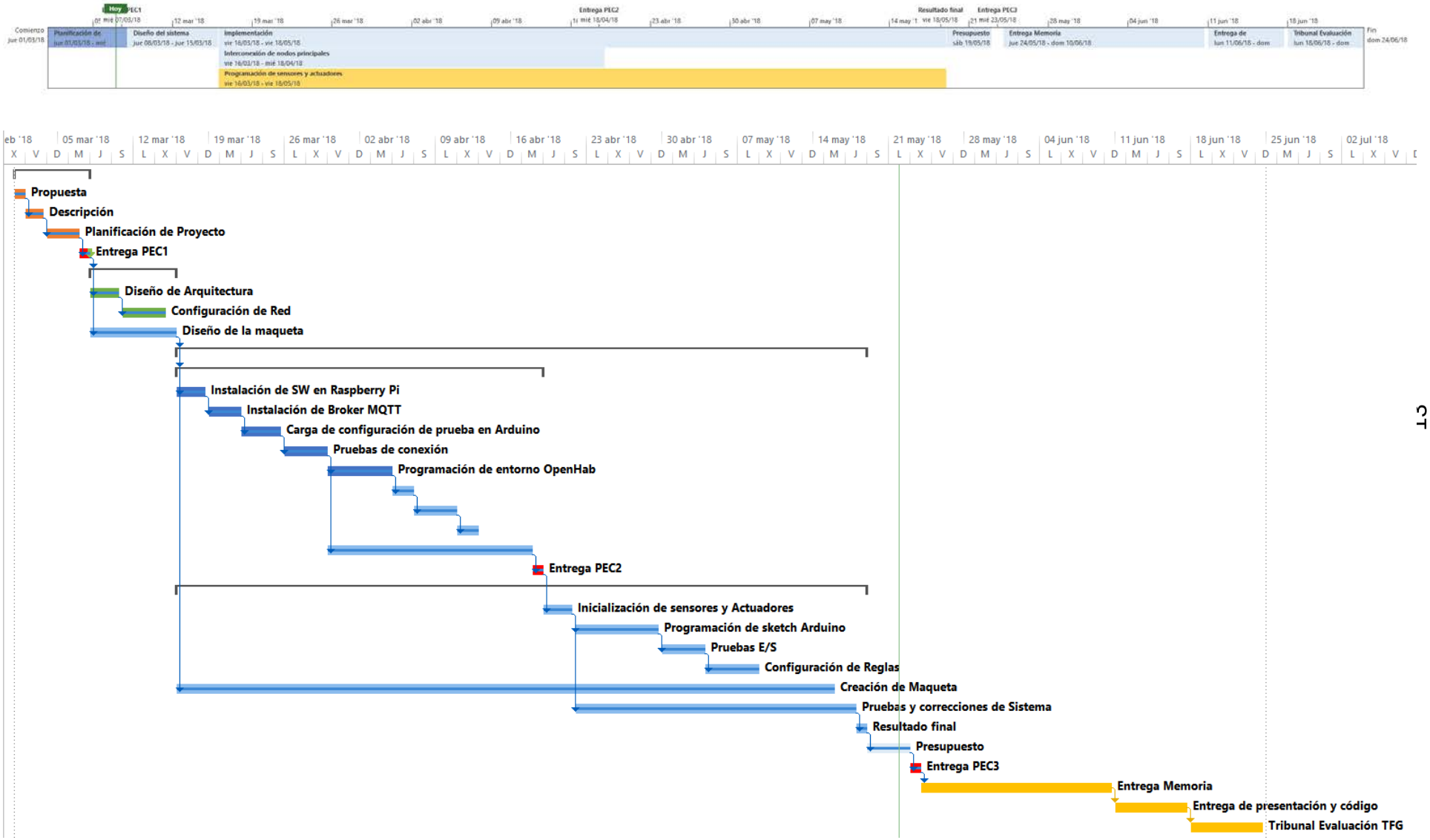
Todo ello se complementará de un video en el cual se realizarán diferentes pruebas del sistema, comprobando y confirmando la utilidad del sistema.

Se busca que todo el desarrollo de este sistema se pueda replicar en comercios, oficinas, o locales de otra índole, permitiendo así la universalización de la solución, aunque los diseñadores particulares deberán adaptarla a sus necesidades específicas. Por ejemplo, dentro de estos grupos, se podría considerar incluir el implementar este sistema en una residencia de ancianos, hospitales psiquiátricos, centros de internamiento de personas con discapacidades… para realizar la monitorización de los internos, activación o desactivación de luces, enchufes… detectores de presencia… y demás automatizaciones que ayudarían al control de los propios centros de esta índole.

1.7 Planificación

Este apartado se ha elaborado a través de la aplicación Microsoft Project 2013, de la que se adjuntan las siguientes capturas de pantalla tanto para la planificación de tareas como para el diagrama de Gantt.

Modo de	Nombre de tarea	Duración	Comienzo	Fin
	Planificación de Proyecto	**7 días**	**jue 01/03/18**	**mié 07/03/18**
	Propuesta	1 día	jue 01/03/18	jue 01/03/18
	Descripción	2 días	vie 02/03/18	sáb 03/03/18
	Planificación de Proyecto	3 días	dom 04/03/18	mar 06/03/18
	Entrega PEC1	1 día	mié 07/03/18	mié 07/03/18
	Diseño del sistema	**8 días**	**jue 08/03/18**	**jue 15/03/18**
	Diseño de arquitectura	3 días	jue 08/03/18	sáb 10/03/18
	Configuración de Red	4 días	dom 11/03/18	mié 14/03/18
	Diseño de la maqueta	8 días	jue 08/03/18	jue 15/03/18
	Implementación	**64 días**	**vie 16/03/18**	**vie 18/05/18**
	Interconexión de nodos principales	**34 días**	**vie 16/03/18**	**mié 18/04/18**
	Instalación de SW en Raspberry Pi	3 días	vie 16/03/18	dom 18/03/18
	Instalación de broker MQTT	3 días	lun 19/03/18	mié 21/03/18
	Instalación de servidor Samba	4 días	jue 22/03/18	dom 25/03/18
	Configuración de Openhab para conectividad	4 días	lun 26/03/18	jue 29/03/18
	Configuración adicional Openhab	6 días	vie 30/03/18	mié 04/04/18
	Instalación de ffmpeg Server	2 días	jue 05/04/18	vie 06/04/18
	Servicios de persistencia (InfluxdB, rrd4j, mapDB)	4 días	sáb 07/04/18	mar 10/04/18
	Grafana	2 días	mié 11/04/18	jue 12/04/18
	Programación de entorno OpenHab	19 días	vie 30/03/18	mar 17/04/18
	Entrega PEC2	1 día	mié 18/04/18	mié 18/04/18
	Programación de sensores y actuadores	**64 días**	**vie 16/03/18**	**vie 18/05/18**
	Inicialización de sensores y actuadores en Arduino	3 días	jue 19/04/18	sáb 21/04/18
	Programación de Sketch Arduino	8 días	dom 22/04/18	dom 29/04/18
	Pruebas E/S	4 días	lun 30/04/18	jue 03/05/18
	Configuración de Reglas	5 días	vie 04/05/18	mar 08/05/18
	Creación de Maqueta	61 días	vie 16/03/18	mar 15/05/18
	Pruebas y correcciones de Sistema	26 días	dom 22/04/18	jue 17/05/18
	Resultado final	1 día	vie 18/05/18	vie 18/05/18
	Presupuesto	4 días	sáb 19/05/18	mar 22/05/18
	Entrega PEC3	1 día	mié 23/05/18	mié 23/05/18
	Entrega Memoria	18 días	jue 24/05/18	dom 10/06/18
	Entrega de presentación y código	7 días	lun 11/06/18	dom 17/06/18
	Tribunal Evaluación TFG	7 días	lun 18/06/18	dom 24/06/18


Propuesta
Descripción
Planificación de Proyecto
Entrega PEC1
Diseño de Arquitectura
Configuración de Red
Diseño de la maqueta
Instalación de SW en Raspberry Pi
Instalación de Broker MQTT
Carga de configuración de prueba en Arduino
Pruebas de conexión
Programación de entorno OpenHab
Entrega PEC2
Inicialización de sensores y Actuadores
Programación de sketch Arduino
Pruebas E/S
Configuración de Reglas
Creación de Maqueta
Pruebas y correcciones de Sistema
Resultado final
Presupuesto
Entrega PEC3
Entrega Memoria
Entrega de presentación y código
Tribunal Evaluación TFG

CAPÍTULO 2 – ESTADO DEL ARTE

El siguiente capítulo está dedicado al estado del arte para que sirva, de aquí en adelante, como base de conocimiento para las diferentes fases del desarrollo del proyecto.

De este modo, se dedicarán los siguientes apartados a definir que es la domótica, estudiar las diferentes tecnologías dominantes en el mercado dedicadas a la automatización del hogar, así como las diferentes topologías existentes y la comparativa entre ellas.

2.1 Domótica

La definición de domótica hace referencia a la capacidad de automatizar una vivienda mediante diferentes sistemas que lo permitan. El término domótica viene de la unión de las palabras domus (casa en latín) y autónomo (del griego: αὐτόνομος; "que se gobierna a sí mismo").

Si se accede a la web de la Asociación Española de Domótica e Inmótica (CEDOM), se obtiene la siguiente definición: *"La domótica es el conjunto de tecnologías aplicadas al control y la automatización inteligente de la vivienda, que permite una gestión eficiente del uso de la energía, que aporta seguridad y confort, además de comunicación entre el usuario y el sistema."*

Los sistemas domóticos son capaces de comunicarse con diversos elementos de una vivienda mediante sus sensores (entradas al sistema) y actuadores (salidas del sistema). Estos elementos pueden utilizarse tanto en el interior como en el exterior de la vivienda, y pueden ampliarse con respecto a nuevas necesidades de los habitantes de la misma, haciendo de un sistema domótico un sistema escalable para adaptarse a dichas necesidades.

Un sistema domótico puede comunicarse con sus sensores y actuadores de diferentes formas. Usando la propia red eléctrica, usando cableado dedicado, mediante tecnología inalámbrica, etc. Además, el sistema domótico debe de comunicarse con las demás redes de la vivienda, como son la red de telefonía, internet o televisión.

Por último, y ya que se ha mencionado a la CEDOM, se expone la definición de inmótica, ya que el trabajo de este proyecto también podrá ser usado en otro tipo de espacios: *"La inmótica es el conjunto de tecnologías aplicadas al control y la automatización*

inteligente de edificios no destinados a vivienda, como hoteles, centros comerciales, escuelas, universidades, hospitales y todos los edificios terciarios, permitiendo una gestión eficiente del uso de la energía, además de aportar seguridad, confort, y comunicación entre el usuario y el sistema."

2.2 Sistemas o protocolos dominantes

Actualmente existen diferentes protocolos para comunicar los elementos del sistema domótico entre sí, como pueden ser X10, KNX, LONworks, ModBUS, etc.

Cada uno de estos sistemas tiene sus propias características, que definen en qué casos es mejor hacer uso de unos u otros. Los principales serían:

Protocolo	Comunicación	Estandar	Detalle	Ventajas	Inconvenientes
KNX	Bus de datos	TCP-IP	Es el sistema más utilizado para domótica	Mayor ancho de banda. Puede usar pasarelas inalámbricas para comunicarse con sensores y actuadores	No es un sistema económico. Recomendado en obra nueva al necesitar cableado dedicado
LONworks			Las redes LONworks compiten directamente con las KNX		
Modbus			Arquitectura maestro/esclavo o cliente/servidor		
X10	Red eléctrica	Pulso de RF de 120 KHz	No está recomendado para la transmisión de video o audio	Sistema económico al no precisar cableado dedicado	Baja velocidad de transmisión
Z-Wave	Inalámbrico	1GHz	Utiliza ondas de radio de bajo consumo que viajan a través de paredes, pisos y armarios		Menor distancia de transmisión (30m)
Zigbee		2,4GHz		Mayor distancia de transmisión (100m)	

Tabla 2 - Diferencias entre protocolos actuales

2.3 Arquitecturas implementables

A la hora de clasificar los tipos de arquitectura o topología a seguir, se pueden crear 3 grupos, al igual que con otro tipo de sistemas de control: arquitectura centralizada, distribuida, o descentralizada.

Sistema	Característica principal	Ventajas	Desventaja
Centralizados	El sistema dispone de un controlador central, el cual se encarga de enviar la información a sensores y actuadores, además de disponer de toda la programación de eventos, condiciones o escenas configuradas en el sistema	Como ventaja, los sensores y actuadores son más simples, por lo que su operatividad e instalación son sencillas, y con un coste de estos dispositivos más reducido que el de otras topologías.	La desventaja de este sistema es que, si el controlador principal falla, el sistema completo deja de funcionar. Además, estos sistemas necesitan cuantioso cableado, puesto que todo el cableado termina en esta unidad principal.
Distribuido	cada sensor o actuador funciona como un propio controlador con capacidad para enviar y recibir información al sistema según lo que recibe de otros dispositivos	cada dispositivo es autónomo por sí mismo, lo que supone que no depende de un controlador central, ni de un posible fallo del mismo, ya que cada elemento tiene implementado acceso al medio (al bus) y la suficiente inteligencia como para ser autónomo sin la necesidad de un "cerebro" central que lo gestione.	necesita bastante programación, y la ampliación de un solo elemento puede ser laboriosa, ya que debe de integrarse en la red del sistema para formar parte del propio sistema.
Descentralizado	En este sistema se dispone de más de un controlador, y todos ellos están conectados entre sí, funcionando como un sistema centralizado en el que cada uno de los controladores gestiona una parte de toda la red de sensores y actuadores	Tiene las ventajas de los sistemas centralizados, pero sin los inconvenientes de caída total del sistema en caso de fallo del nodo central	Estos sistemas comparten la desventaja de necesitas cuantioso cableado

Tabla 3 - Tipos de Arquitecturas implementables

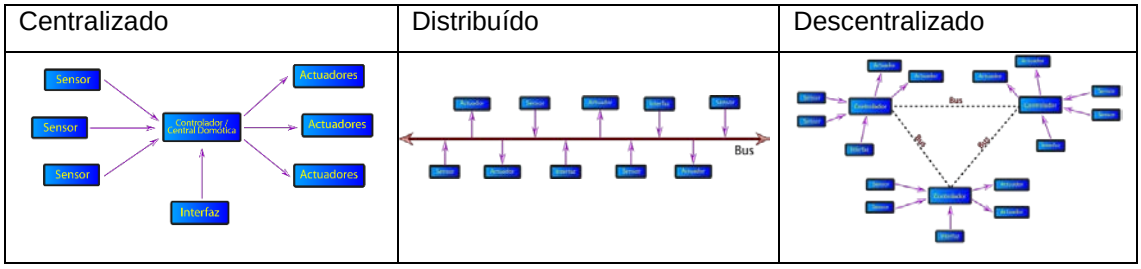


Figura 3 - Arquitecturas Domóticas

CAPÍTULO 3 - DISEÑO

3.1 Diseño del sistema

En este apartado se identifican los aspectos que se consideran necesarios para el sistema.

Lo primero se analiza serán los elementos que compondrán el sistema domótico. Se usarán 2 tipos de placas controladoras: Arduino – para recibir información de los sensores, y enviar órdenes a los actuadores – y Raspberry Pi, la cual tendrá la responsabilidad de consolidar toda la información que reciba de los Arduino y tomará las decisiones según los parámetros configurados. Esto se ha definido así, porque permite lanzar una orden a un Arduino por un parámetro recibido a través de otro diferente, por lo que se obtendrá una red homogénea en el que todos los elementos colaboran entre sí.

Para realizar la comunicación entre ambos dispositivos se utilizará la plataforma Opensource “OpenHAB”, ya que es directamente compatible con el cerebro del sistema, la Raspberry Pi, y además dispone de múltiples interfaces (Bindings) para controlar diferentes dispositivos con conexión a la red, lo que supone un paso más en la domotización del hogar.

Para realizar la comunicación entre los diferentes dispositivos, Arduino y Raspberry, se hará uso del protocolo MQTT, ya que se trata de un protocolo muy liviano, y que tanto nuestros Arduino, como Openhab, son capaces de interactuar con este.

Para poder disponer de servicios de BBDD, el sistema se apoyará en InfluxDB, una BBDD Open Source que proporciona la potencia necesaria para registrar todos los eventos que se producen en el sistema, y además permite una comunicación directa con Grafana, un sistema que permite realizar gráficas de todo aquello que ocurre sobre el sistema domótico.

Como se considera vivienda de obra nueva, toda la red será cableada, aunque existen dispositivos como el Esp01, o el esp8266, que permitirían establecer puntos Wireless en el hogar en aquellos lugares donde no llegasen los cableados, o incluso interfaces Wifi para Arduino, con lo que la comunicación sería inalámbrica haciendo uso de los mismos equipos.

El motivo de la realización de la red cableada se centra en la robustez de las comunicaciones cableadas, aunque el coste de implementarlas es mayor que el de las inalámbricas.

3.1.1 Diseño de la arquitectura

La arquitectura del sistema seguirá un concepto de estrella, en el que se parte de un nodo principal en el que se ubicará la Raspberry Pi, hacia los nodos secundarios en los que se instalarán los Arduino, y de la misma forma entre los Arduino y sus sensores y actuadores.

Se ha decidido usar este tipo de infraestructura para permitir que cada elemento se conecte por si solo a la red local, por lo que facilita la introducción de nuevos nodos en el sistema, o el apagado de uno de ellos sin que afecten al resto (excepto en la ejecución de posibles reglas vinculadas, las cuales habría que modificar)

El esquema a nivel de equipamiento de control sería el siguiente:

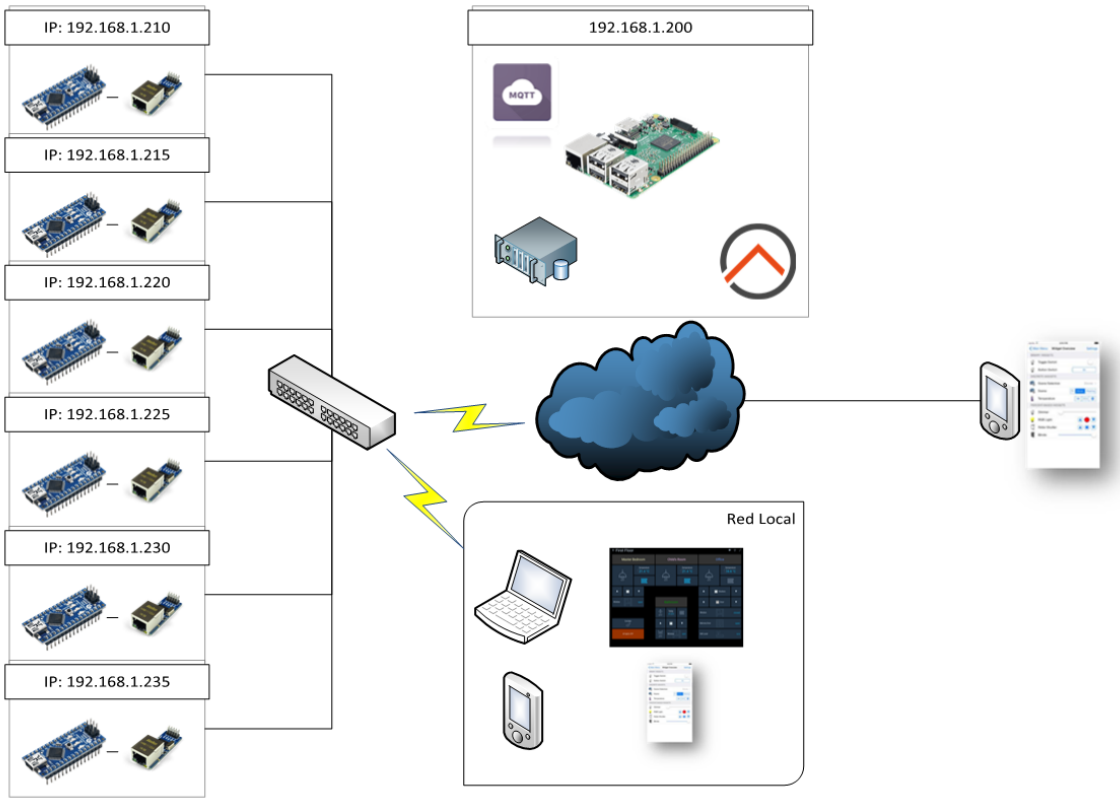


Figura 4 - Esquema Equipamiento de control

Donde es necesario tener en cuenta que la red completa podrá identificarse mediante las 7 capas del modelo OSI:

Capa	Nombre	Servicio
7	Capa de Aplicación	MQTT
6	Capa de Presentación	
5	Capa de Sesión	
4	Capa de Transporte	TCP
3	Capa de Red	IPv4
2	Capa de enlace	Ethernet / WIFI
1	Capa física	

Figura 5 - Capas OSI del sistema

3.1.2 Configuración de Red

Se ha realizado una división de red tomando como partida la dirección IP que se asignará al nodo principal, a la Raspberry Pi, la 192.168.1.200. Esto es debido a que el rango de direcciones asignadas por DHCP a los equipos de la red por parte del Router estará definido entre 192.168.1.2 y 192.168.1.199. De esta forma, se evitan posibles conflictos de IP.

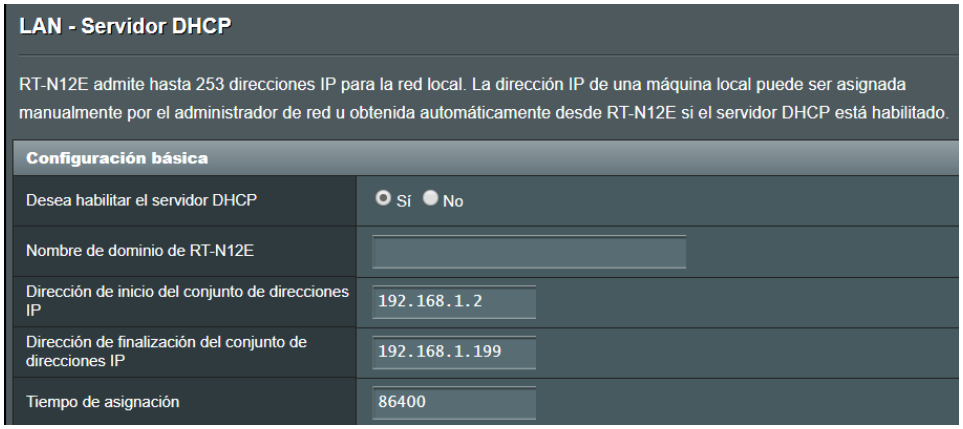


Figura 6 - Configuración de Red

El direccionamiento del sistema domótico será el siguiente:

Dispositivo	Servicio	IP	Puerto
Raspberry Pi	OpenHAB	192.168.1.200	8080
	MQTT	192.168.1.200	1883
	InfluxDB	192.168.1.200	8086
	Grafana	192.168.1.200	3000
	IP Cam - Python	192.168.1.200	8000
Arduino 1	Hab Principal	192.168.1.210	
Arduino 2	Salón	192.168.1.215	
Arduino 3	Cocina	192.168.1.220	
Arduino 4	Hab Invitados	192.168.1.225	
Arduino 5	Baño	192.168.1.230	
Arduino 6	Pasillo	192.168.1.235	

Tabla 4 - Configuración de Red

Se puede apreciar que se dispone de diferentes puertos en la misma dirección IP de la Raspberry. Esto es debido a que se hará uso de los puertos virtuales de forma diferenciada para acceder a los diferentes servicios en cada uno.

3.1.3 Diseño de la maqueta

La maqueta del sistema constará de 2 partes diferenciadas:

1. Un modelo impreso en plástico de la planta baja, en la cual estarán ubicados sensores y leds a modo de testers para probar los actuadores de la vivienda. En el código real de una vivienda habría diferencias, puesto que donde se ubiquen los leds realmente habría relés, pero son cambios menores.

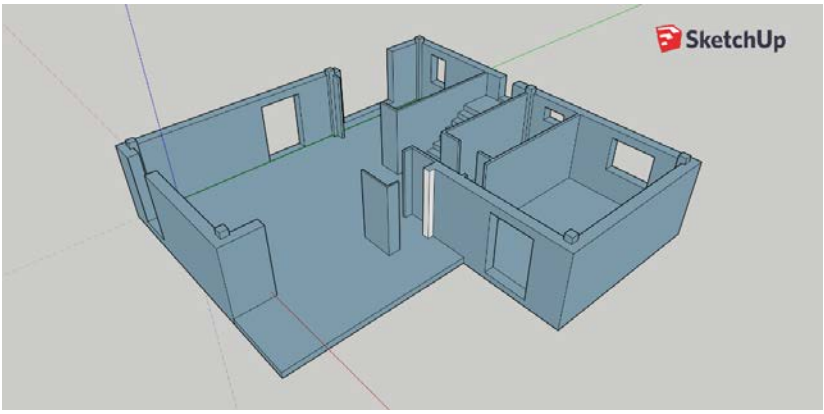


Figura 7 - Maqueta SketchUp

2. Un modelo plano simulando el dormitorio principal (planta alta), para probar código real. En este modelo se incluyen varios interruptores mecánicos para ver cómo se podría adaptar un sistema domótico a una vivienda sin la necesidad de realizar el cambio de todos los interruptores del hogar, aunque si habría que hacer cambios en parte para cambiar algunos conmutadores por cruzamientos, pudiendo reaprovechar dichos conmutadores para incluirlos en estancias con interruptores simples.

3.1.3 Funcionamiento básico del sistema

El principio básico de funcionamiento del sistema es la interconexión de los diferentes equipos mediante el protocolo de comunicación MQTT, el cual se localiza en las capas superiores de OSI, y que se apoya en una red TCP/IP (Capas 4 y 3).

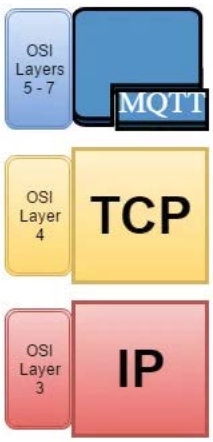


Figura 8 - Capas OSI

Esto quiere decir que toda la interconexión estará centrada en la conexión de los dispositivos mediante la red local, ya sea cableada o inalámbrica.

El bróker MQTT estará instalado en Raspberry Pi, y será el encargado de procesar las llamadas de los clientes del sistema, en este caso cada uno de los Arduino y OpenHab, que usarán Topic definidos para la comunicación inequívoca de las comunicaciones.

Entonces, por un lado, en cada Arduino se encontrarán conectados una serie de sensores y actuadores, programados en la propia lógica de Arduino, y por el otro, Openhab dispone de una lógica de Items. Se puede afirmar que cada sensor o actuador tendrá su propio Item en OpenHab.

Solo cabe añadir que cada sensor/actuador – Item, tendrá un identificador único en el sistema, un Topic único en el lenguaje MQTT.

La representación de cada Item en el sistema se realiza a través de una interfaz en Openhab, la cual se denomina Sitemap. Además, para realizar la configuración de reglas, también se apunta a cada uno de los Items y a su estado en un determinado momento. Para finalizar, el sistema dispone de un servicio de persistencia de datos, InfluxdB, y un servicio de representación de los mismos, Grafana, con lo que el esquema general sería el siguiente:

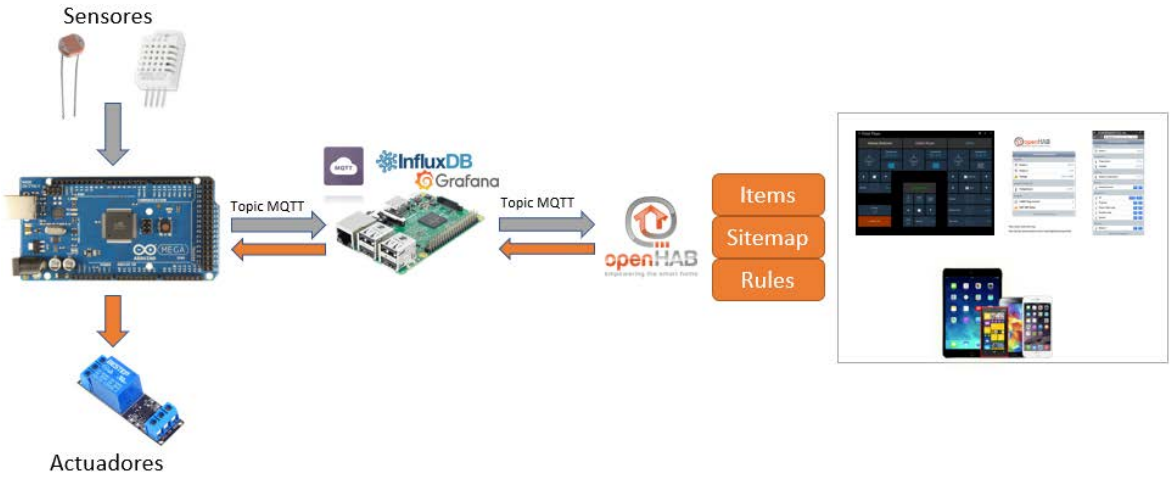


Figura 9 - Esquema Completo de la solución

3.2 Tecnologías y componentes utilizados

3.2.1 Arduino

En este proyecto se usará una plataforma Hw de código abierto, basada en una placa con salidas y entradas digitales y analógicas, que será donde se conectarán los sensores y los actuadores del sistema.

Existen diferentes placas que difieren en forma, tamaño y prestaciones dentro de la familia Arduino, como pueden ser el Arduino Mega, Uno o Nano, que pueden utilizarse en el proyecto, pero en este caso se usará el Arduino Mega.

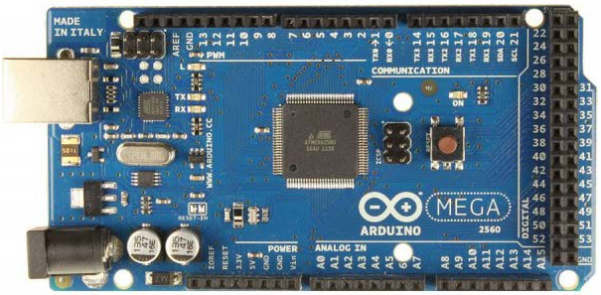


Figura 10 - Arduino Mega

El Arduino Mega es un dispositivo de la familia Arduino, basado en el chip ATmega1280, y que ofrece mayor capacidad que sus hermanos en cuestión de conectividad, pero el factor determinante para su uso es su mayor capacidad de procesador, para gestionar de manera eficiente la bidireccionalidad del protocolo MQTT. Sería posible usar un Arduino Nano, o Uno, en su lugar, pero el dispositivo estaría más limitado en cuestión de código y entradas y salidas. Sus características principales son:

Microcontroller	ATmega1280
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	128 KB of which 4 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Tabla 5 - Arduino Mega

Debido a que dicho dispositivo no dispone de conectividad Ethernet de serie, se instalará una placa de ampliación denominada Ethernet Shield (Escudo Ethernet) compatible, en nuestro caso el W5100:

Figura 11 - W5100

3.2.2 Sensores

Como se ha definido en la propuesta, se obtendrán datos de luminosidad, temperatura y humedad, además de diferentes sensores para monitorizar eventos de alarma como sensores de movimiento, fuego, gas o inundación, por lo que usaremos los sensores adecuados para obtener esta información:

3.2.2.1 Temperatura y humedad

Para realizar las mediciones de temperatura y humedad se hará uso del sensor DHT22. También existe el DHT11, pero es mucho menos preciso, y para realizar las mediciones en la vivienda es importante que lo sea, sobre todo porque activará la calefacción cuando la temperatura en la estancia baje de 18º. Si el sensor no es preciso, existe el riesgo de cambios bruscos en la medición de Tª, pudiendo suponer el arranque y parada constante de la caldera, con el sobreconsumo y riesgo de avería que esto conlleva.

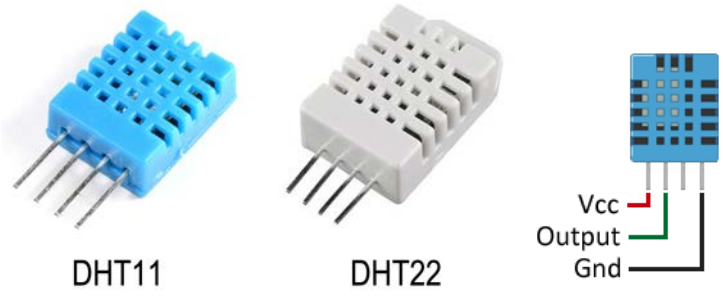


Figura 12 - DHT11, DHT22, y conexiones DHT22

3.2.2.2 Luminosidad

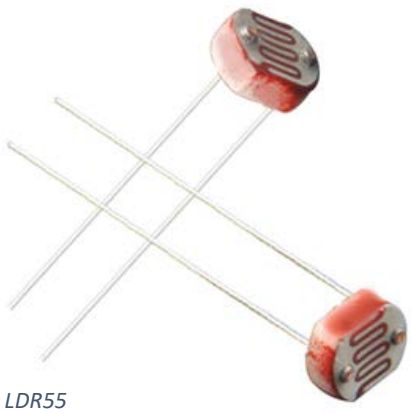


Figura 13 - LDR55

Para realizar la medición de luminosidad se utilizará una fotorresistencia LDR (light-dependent resistor). El funcionamiento de esta fotorresistencia consiste en que varía su resistencia en función de la luz recibida. Entonces, conectándola a una de las entradas analógicas de nuestro Arduino correspondiente, podrá estimar el nivel de luz.

3.2.2.3 PIR, o sensor de movimiento

Figura 14 - PIR

Los sensores PIR parten de la base de que los seres vivos desprenden calor, el cual se emite en forma de radiación infrarroja que puede ser detectada con los dispositivos adecuados.

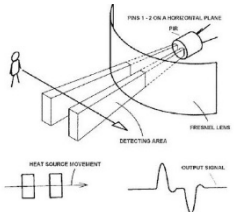


Figura 15 - Funcionamiento PIR

Los sensores PIR son elementos capaces de detectar cambios en la radiación infrarroja que reciben entre ambos planos, y disparan una alarma al percibirlo.

3.2.2.4 Sensor de llama

El sensor de llama es un sensor que tiene la capacidad de detectar la existencia de combustión debido a la presencia de luz emitida por la misma.

El espectro de emisión de llama depende de los elementos que intervienen en la reacción, aunque las longitudes de onda más comunes abarcan entre 185nm-260nm dentro del espectro ultravioleta y de entre 4400-4600nm dentro del espectro infrarrojo.

El sensor que se usará en el proyecto, según su propio Datasheet, incorpora una placa de medición estándar con un comparador LM393, con el siguiente gráfico de respuesta:

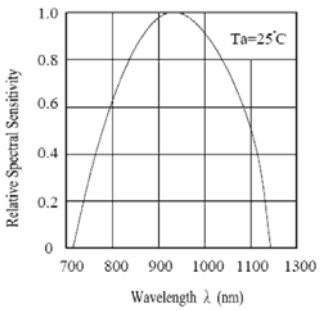


Figura 16 - Sensor de Llama

Puede observarse que, aunque funciona correctamente en la detección de llama, ha de configurarse de forma muy precisa para no incurrir en falsos positivos, aunque eso suponga retrasar la detección de fuego a que la llama sea más intensa, o que se encuentre cerca del lugar de la instalación del sensor.

3.2.2.5 Sensor de Gas

Los sensores de gas son sensores analógicos, por lo que son sencillos de implementar. Necesitan una fuente de 5V para el calentador (necesitan unas 48h conectados antes de estabilizarse las medidas)

Figura 17 - Sensor de Gas MQ135

Su base de funcionamiento es electroquímica, ya que varían su resistencia cuando se exponen a determinados gases; posee un calentador encargado de aumentar la temperatura interna y con esto el sensor pueda reaccionar con los gases provocando un cambio en el valor de la resistencia. Este valor se leerá desde Arduino y se mapeará entre unos valores determinados para representar el nivel del gas en cuestión que se está monitorizando

3.2.2.6 Sensor de Inundación

El sensor de inundación en Arduino tiene un funcionamiento básico, usando las propiedades conductoras del agua. Cuando aumenta el nivel de agua, realiza un cortocircuito entre las bandas verticales del sensor y hace variar el nivel de resistencia. Una vez calibrado se podrá conocer la presencia o no de agua sobre el mismo sin dificultad.

Figura 18 - Sensor de Inundación

3.2.2.7 Otros sensores

Existen diferentes sensores que podrían adaptarse a este sistema como pueden ser sensores de proximidad, de contacto, de rotura de cristales, de presión… Todos ellos podrían ser integrados en este sistema para poder monitorizar cualquier evento que se precise, ya sea mediante uno o más sensores en paralelo.

3.2.3 Actuadores

Debido a que se usará la información disponible de la temperatura del hogar para accionar la calefacción del hogar, se considera la utilización de radiadores eléctricos en las habitaciones, por lo que se realizará su control mediante placas de relés. Además, se instalarán leds y un buzzer para visualizar eventos sin hacer uso de la app:

3.2.3.1 Relés

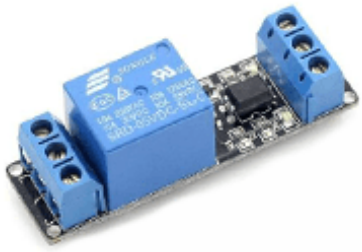


Figura 19 - Relé de 5V

Estos dispositivos nos permiten controlar una carga de 220V a través de una salida de control de Arduino de 5V, y nos permiten manejar cargas de hasta 10A, por lo que además de los radiadores, cubriría casi cualquier elemento doméstico que se necesite domotizar.

Físicamente, un relé se comporta como un interruptor convencional, ya que el circuito primario se conecta con la electrónica de baja tensión, en nuestro caso Arduino, y recibe la señal de encendido o apagado, y el circuito secundario es el interruptor encargado de encender o apagar la carga. El tiempo de conmutación de esta carga es de unos 10ms, por lo que el resultado de la activación o desactivación del circuito es casi instantáneo.

3.2.3.2 Buzzer

Técnicamente, un buzzer es un transductor electroacústico, ya que convierte señales eléctricas en sonido, al igual que un altavoz.

Estos dispositivos son transductores piezoeléctricos, los cuales tienen la propiedad de variar su volumen al ser atravesados por corrientes eléctricas, haciendo vibrar una membrana al atravesar dicho material con una señal eléctrica.

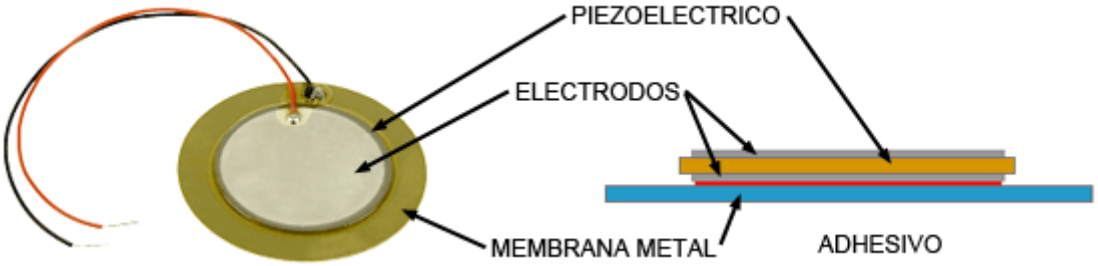


Figura 20 - Funcionamiento Buzzer

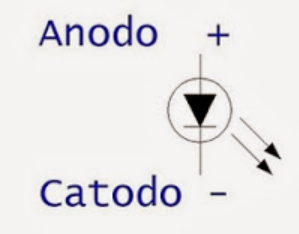
3.2.3.3 Diodos led

El diodo LED es un componente capaz de transmitir luz, la cual se usará para simular diferentes dispositivos en la maqueta. Dicha luz es emitida cuando la corriente eléctrica lo atraviesa, y, además, a más intensidad de corriente, más luz. Por ello es necesario incluir una resistencia en serie para proteger el dispositivo. Generalmente se inducirá un voltaje desde Arduino de 3,3V (Señal HIGH), por lo que se podrá proteger con una resistencia en serie de 220Ω, de forma que la corriente que lo atraviese ronde los 15mA.

Figura 21 - Diodo Led

$$I = \frac{V}{R} = \frac{3.3}{220} = 15mA$$

Para su funcionamiento, puesto que se trata de un diodo, ha de tenerse en cuenta que debe de estar polarizado en directa, es decir, al ánodo (terminal positivo y pata más larga) ha de conectarse a 3,3V, y el cátodo, a GND.



3.2.3.4 Otros actuadores

Aunque no se han contemplado en este proyecto debido a la necesidad de síntesis, es posible añadir válvulas solenoides, que nos permiten cortar el gas o el agua en caso de inundación o escape de gas, o motores, que nos permiten accionar, por ejemplo, persianas o puertas de garaje desde nuestro sistema.

3.2.4 Raspberry Pi

Figura 22 - Raspberry Pi 3 B

La Raspberry Pi es un ordenador de placa reducida, menos potente que un ordenador convencional, pero suficiente para procesar las necesidades domóticas de una vivienda. Funciona con un SO adaptado de Debian, denominado Raspbian, pero permite usar diferentes sistemas operativos. Se usará en el proyecto la última versión que existe en la actualidad, el modelo 3 B, que dispone de un procesador 1.2GHz 64-bit quad-core ARMv8, 4 USB, puerto Ethernet, Wifi, Bluetooth, y 17 GPIO, que son entradas y salidas que pueden controlar sensores o actuadores que se instalen cerca del emplazamiento del controlador, evitando así la instalación de un Arduino adicional.

Este dispositivo será el encargado de ejecutar OpenHAB en su sistema operativo, y recibirá los datos de todos los Arduino instalados en la vivienda por medio del protocolo MQTT. Por otro lado, enviará órdenes a los actuadores a través de estos mismos Arduino usando el mismo protocolo.

3.2.5 OpenHAB

OpenHAB es un software desarrollado en Java que pretende integrar múltiples sistemas de domótica que permiten automatizar distintas funciones de la Smart home.

El diseño visual es bastante amigable y moderno, y dispone de un sistema responsivo, en el que es posible adaptar los elementos de la plataforma a las necesidades de cada vivienda o local.

Figura 23 - Pantalla OpenHab

OpenHAB usa Java como lenguaje de programación. Para su programación se utilizará Notepad++

El funcionamiento de trabajo de OpenHab se basa en la programación de Items (objetos virtuales) que representan Things (cosas reales) dentro de un Sitemap (el mapa de la propia interfaz del sistema).

Así, se ha de crear una conexión entre cada dispositivo del sistema (sensor, actuador…) con un Item inequívocamente asignado. De esta forma OpenHab puede tanto conocer el estado de un sensor determinado, como enviar una orden a un actuador.

El modo de comunicación entre un objeto real y un Item se produce mediante un enlace con el protocolo MQTT.

3.2.6 Protocolo MQTT

El protocolo MQTT se basa en un nodo central (En este caso Raspberry Pi) que hace de servidor o "bróker" con una capacidad de hasta 10000 clientes. Este gestiona la red y la transmisión de los mensajes.

La comunicación sobre este protocolo es simple y ligero, empleando pocos recursos:

- Para mantener activo el canal, los clientes mandan periódicamente un paquete (PINGREQ) y esperan la respuesta del bróker (PINGRESP).
- La comunicación está basaca en "topics", sobre los cuales el cliente emisor publica el mensaje y los clientes receptores que deseen recibirlo deben subscribirse a él. Un cliente podrá ser al mismo tiempo emisor y receptor, por lo que 2 dispositivos pueden comunicarse sobre el mismo Topic, para lo cual han de tener un mensaje codificado que ambos entiendan. Por ejemplo, enviar un 1 puede significar ON, y un 0 OFF, o incluso asignar un estado o un mensaje a cada carácter emitido. De esta forma, enviando muy pocos datos, se transmite la información necesaria para el correcto funcionamiento del sistema.
- La comunicación puede ser de uno a uno, o de uno a muchos.
- Un "topic" se representa mediante una cadena y tiene una estructura jerárquica, usando / para separar los saltos que el mensaje debe de dar en la cadena
    - El diseño de red MQTT de este proyecto será el siguiente:

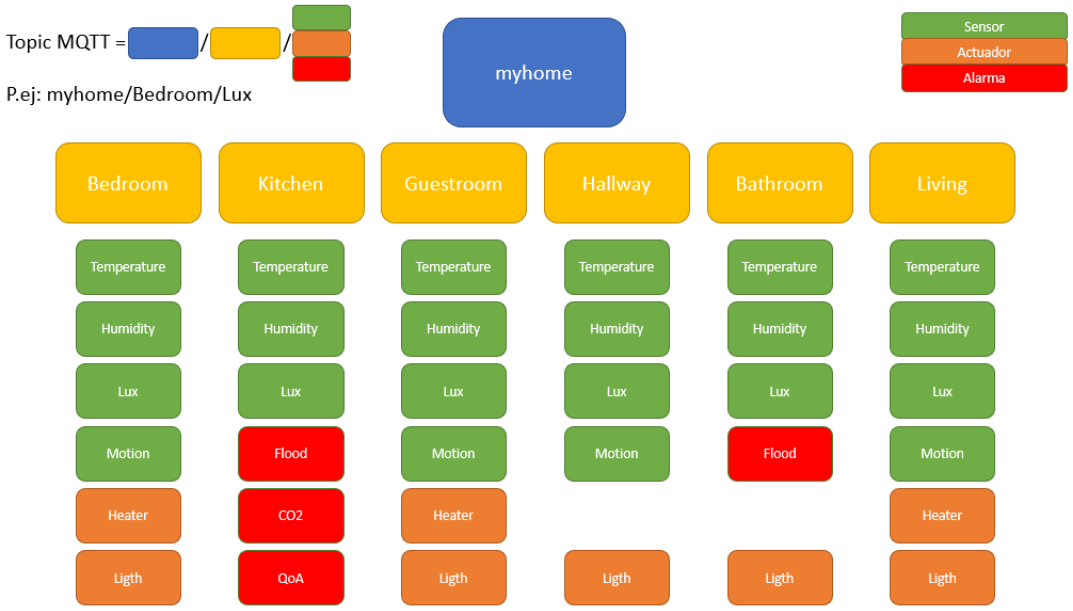


Figura 24 - Esquema MQTT del sistema

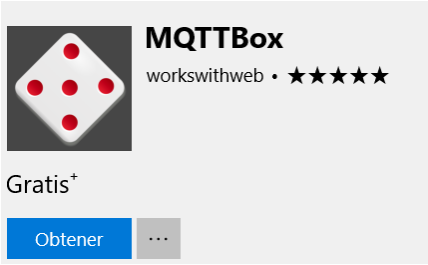
CAPÍTULO 4 - IMPLEMENTACIÓN

4.1 Interconexión entre los nodos principales

El primer paso, una vez diseñada la estructura que dispondrá el proyecto, será el de proveer a la Raspberry Pi de los servicios necesarios para poder desarrollar el sistema domótico. Se empezará con la instalación del sistema operativo Raspbian, seguido por el programa Open Source OpenHAB. A partir de este punto se instalarán además los diferentes servicios como el bróker MQTT, los servicios de persistencia InfluxDB, rrd4j, o mapDB. Por último, también se incluirán los servicios de Grafana, útiles para realizar gráficas más avanzadas que las propias que permite OpenHAB, y se creará un código en Python, que se usará para poder hacer streaming de una cámara de seguridad como sistema de videovigilancia. La guía paso a paso de instalación y preparación del sistema se encuentra en los anexos.

4.1.1 Pruebas de conexión OpenHAB—MQTT

Llegados a este punto, Openhab ya puede conectarse con el bróker MQTT bidireccionalmente, por lo que podrán realizarse pruebas de conexión. Para ello, puede hacerse uso de cualquier programa que lea mensajes de MQTT, por ejemplo, MQTTBox, gratuito para Windows, pero serviría cualquiera similar:



Se instala y se accede con el cliente al servidor con la configuración que se ha usado antes, y se prueba la conexión con el Broker:

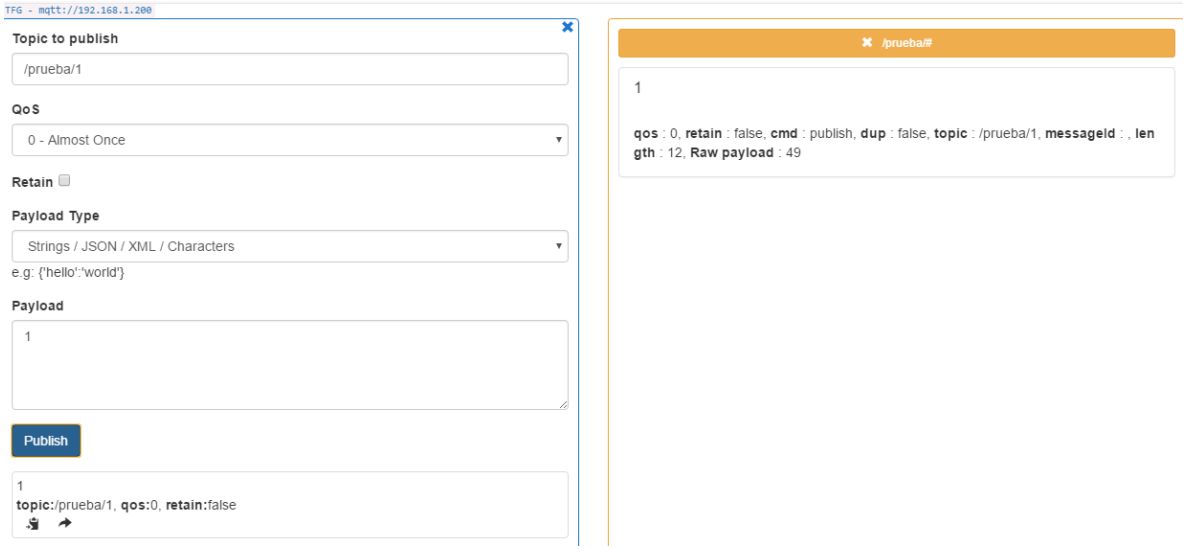


Figura 25 - Prueba conectividad MQTT

Lanzando un 1, se recibe un 1 en la escucha. Se prueba a lanzar un comando similar con Openhab directamente. Para ello se crea un Item en un fichero .items, y se habilita en un .sitemap de prueba:

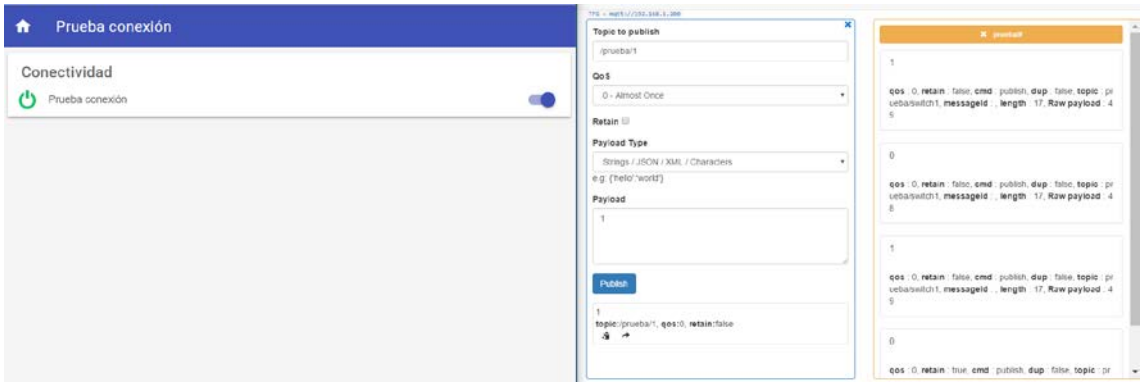
Prueba.items

```
Switch Prueba_MQTT "Prueba_MQTT" {mqtt="">[broker:prueba/switch1:command:ON:1],>[broker:prueba/switch1:command:OFF:0],<[broker:prueba/switch1:state:ON:1],<[broker:prueba/switch1:state:OFF:0]"}
```

Prueba.sitemap

```
sitemap Prueba_MQTT label="Prueba conexión" {
        Frame label="Conectividad" {
                Default item=Prueba_MQTT label="Prueba conexión"
}
}
```

Y se obtiene lo siguiente:



Se puede apreciar que funciona perfectamente. En el momento que se enciende el switch se recibe un 1, y si se apaga, un 0.

4.1.2 Configuración adicional Openhab

Una vez se dispone de conectividad de Openhab con la Red local, y con el bróker MQTT, se procede a instalar diferentes servicios que aportarán valor al sistema domótico del hogar. Para este trabajo se ha optado por:

- Astro Binding: Para obtener valores relacionados con las órbitas del sol y la luna. Este Binding proporcionará información sobre la hora de la salida y la puesta de sol, radiación solar, el estado de la luna…
- Samsung Binding: Como se dispone en el hogar de un televisor Samsung Smart TV, se conectará con el sistema domótico para hacer uso de sus funcionalidades a través de la red local
- Systeminfo Binding: Proporcionará información sobre el uso del servidor
- Yahoo Weather Binding: Proporcionará información del tiempo vía internet
- OpenHAB Cloud: OpenHAB Cloud es un servicio de OpenHAB que habilita el acceso al sistema desde el exterior del hogar.

4.1.3 Programación de entorno OpenHAB

4.1.3.1 Configuración

Una vez se han creado y configurado todos los servicios necesarios para el sistema domótico, se procederá a crear el entorno visual que tendrá al acceder desde aplicación o vía web. El entorno del sistema OpenHAB se estructura en varias partes, configurables en parte mediante ficheros y en parte mediante el portal PaperUI:

- Things:

Las *cosas* (Things) son las entidades que se pueden agregar físicamente al sistema y que potencialmente pueden proporcionar varias funcionalidades. Las *cosas* no solo tienen que ser dispositivos, sino que además pueden representar un servicio web o cualquier otra fuente manejable de información y funcionalidad. Desde la perspectiva de usuario, son relevantes para el proceso de configuración, pero no para la operación.

Las *cosas* pueden tener propiedades de configuración, que pueden ser opcionales u obligatorias. Tales propiedades pueden ser información básica como una dirección IP, un token de acceso para un servicio web o una configuración específica del dispositivo que pueda alterar su comportamiento

En general, las “cosas” se importarán a través del menú Paper UI, y sobre las cuales se crearán los “Items”.

- Items:

En OpenHAB, los Items representan todas las propiedades y capacidades de la domótica del usuario. Estos elementos establecen una separación estricta entre el mundo físico (las *cosas*) y la aplicación, que se basa en la noción de *elementos* o Items (también llamada capa virtual).

Mientras que un dispositivo o servicio puede ser bastante específico, los artículos son sustituciones unificadas dentro del mundo OpenHAB. Los elementos pueden ser cadenas, números, conmutadores o alguno de los varios tipos básicos de elementos disponibles.

Una característica única de estos Items es la capacidad de conectarse al mundo exterior a través de enlaces. Un item no almacena simplemente la información establecida por el software (por ejemplo, DESACTIVADO, 3.141 o "Sin error"), sino que la información almacenada por un item también puede establecerse mediante acciones que tienen lugar en el hogar.

También existen los llamados “Group Items”, los cuales reúnen otros Items en grupos. Estos grupos de elementos pueden ser a su vez miembros de otros elementos grupales. Las interfaces de usuario pueden mostrar elementos de grupo como entradas individuales, los cuales permiten realizar acciones directas sobre un grupo de elementos (por ejemplo, apagar directamente todas las luces del hogar).

En este proyecto se han definido, una serie de Items que se pueden ver en el fichero adjunto **Items.items**

El formato de los Items es el siguiente [https://docs.openhab.org/configuration/items.html]:

```
itemtype itemname "labeltext [stateformat]" <iconname> (group1, group2, ...) ["tag1", "tag2", ...] {bindingconfig}
```

Por ejemplo:

```
Switch Kitchen_Light "Kitchen Light" {mqtt="<[...], >[...]" }
String Bedroom_Sonos_CurrentTitle "Title [%s]" (gBedRoom) {channel="sonos:..."}
```

- Sitemaps:

En OpenHAB, una colección de cosas y objetos representa objetos físicos o lógicos en la configuración de domótica del usuario. Los sitemaps se utilizan para seleccionar y preparar estos elementos a fin de componer una presentación orientada al usuario de

esta configuración para varias interfaces de usuario (UI), incluyendo la interfaz BasicUI, y las diferentes apps disponibles.

Cada Sitemap que se crea puede verse directamente en la aplicación móvil, o a través del menú Basic UI. En este Sitemap se representan también los Items que se han creado anteriormente. Se adjunta fichero **Sitemap.sitemap** con los elementos que se han creado en este proyecto.

Dentro de estos Sitemaps se representarán además gráficos que usen los servicios de persistencia instalados en el sistema. En la habitación principal, además, se mostrará un gráfico que usa Grafana como motor, haciéndolo más atractivo que los gráficos por defecto de OpenHab.

Por último, se dispondrá de un submenú en el que se podrá observar una cámara de seguridad, la cual será la propia cámara de Raspberry Pi, enviando imágenes por medio de http mediante el un proceso en Python configurado a través de la dirección local http://192.168.1.200:8000

- Rules:

Las "Reglas" se utilizan para automatizar procesos, y se escriben en un lenguaje basado en Java, Xtend: cada regla se puede configurar para invocar una secuencia de comandos que realice cualquier tipo de tareas, como encender las luces al entrar en casa al ser detectado por un sensor de movimiento, activar una alarma con estos mismos sensores de movimiento, o además exigir que si está activada envíe una notificación al móvil con la detección de intrusión, iniciar temporizadores... y casi cualquier automatización entre todos los elementos del sistema.

4.1.3.2 Interfaz

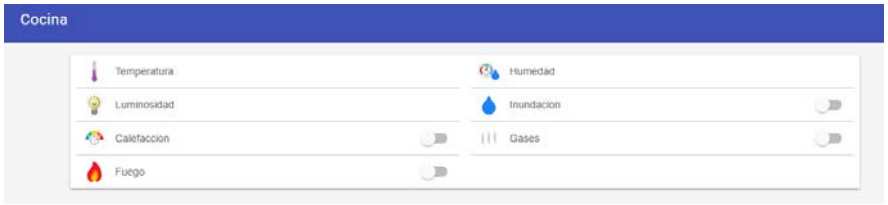
Openhab dispone de 2 interfaces:

- Basic UI:

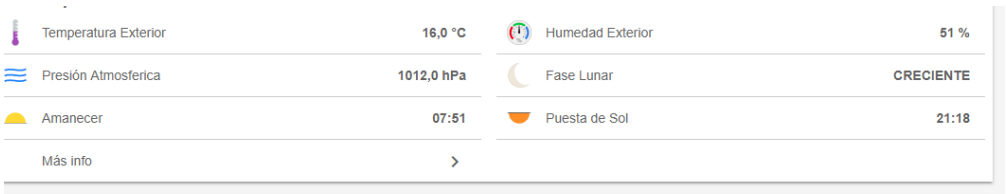
Se trata del interfaz principal del sistema, el cual muestra los datos que se han configurado en los ficheros de Sitemaps e Items

Figura 26 - OpenHab - Basic UI

Y dentro de cada uno de estos menús, se encontrarán los ítems creados dentro de estos grupos:



También se han representado menús que se alimentan solo de la información de los Bindings, como el siguiente de Astro Binding y Yahoo Weather Binding:



E incluso un apartado para el televisor:

- HABPanel:

HABPanel es una interfaz de Openhab pensada para ser gestionada desde una Tablet y controlar los elementos de la vivienda dentro de la red local. Aunque es posible el acceso desde el exterior, su visualización está pensada para ser ejecutada en un equipo local (PC, Tablet…). Su configuración es posible realizarla desde la propia interfaz, o por código. Se adjunta **HABPanel.json**

Algunas de las pantallas que se usarán en el proyecto son las siguientes:

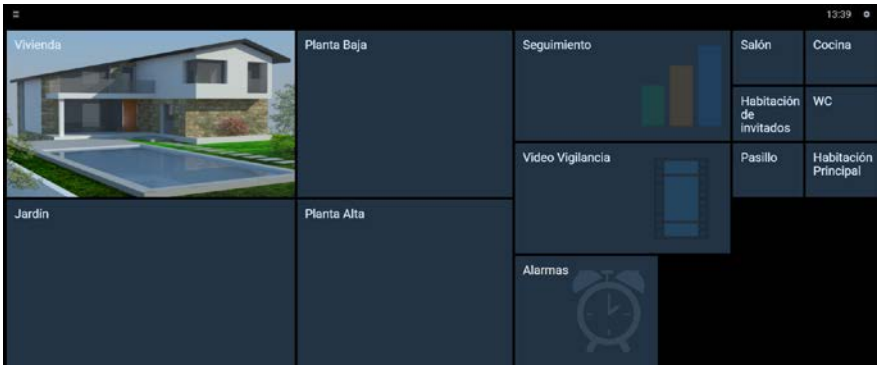
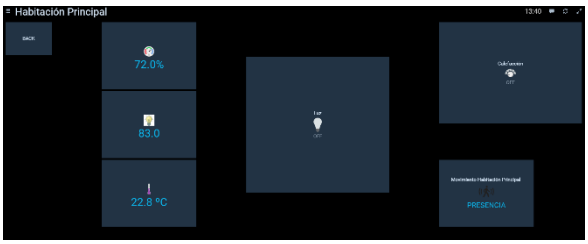


Figura 27 - OpenHab - HabAdmin



Se pueden crear diferentes Dashboard (Tablones), sobre los cuales representar la información. Es recomendable crear varios dependiendo del tipo de dispositivo desde el que se va a realizar la conexión, puesto que el cambio de resolución de los dispositivos afecta a la visualización y han de ser optimizados.

4.2 Inicialización y configuración de sensores y actuadores en Arduino

Una vez realizada la configuración del nodo central, se ha de programar cada uno de los nodos secundarios alojados en cada Arduino MEGA. Por motivos de espacio se tratará el código de 2 de estos nodos que incluyen el código general que tendrían el

resto de los nodos. Habitación principal y Simulación. Sobre este último se implementará código para simular los elementos del hogar, tanto de las estancias como los relacionados con la activación y desactivación de la alarma por medio de RFID.

4.2.1 Arduino – Código genérico

Para realizar la configuración de Arduino, se diferenciarán las siguientes partes:

- Inicio: En la primera parte de código se realiza la inicialización y declaración de variables globales. Entre estas variables están, además de la inicialización de los ítems relacionados con los sensores y actuadores, la configuración de red que se cargará en Arduino para que se conecte tanto a la red local como al bróker MQTT.
- Funciones: También en esta parte inicial del código, existe la posibilidad de crear diferentes funciones, que a su vez también pueden contener variables locales. Estas funciones se usan para realizar tareas concretas dentro del código, para conseguir que las diferentes utilidades del código funcionen de forma independiente.
- Función setup(): En esta parte del código se realiza la carga de la configuración general. Es la primera parte del código que realmente se ejecuta, y este se ejecuta una sola vez en todo el tiempo de ejecución.
- Función loop(): Esta función contiene la parte del código que se ejecutará continuamente (lectura de entradas, activación de salidas…). En el caso de un programa sencillo, lo habitual es realizar la carga dentro de la función loop() directamente. En el caso de proyectos más complejos, se lanza desde esta función las diferentes funciones necesarias para la ejecución del programa.

Figura 28 - Programación Arduino

4.2.2 Uso de Ethernet Shield con Arduino Mega

Para poder lanzar la conexión de Arduino con la red local, se tiene que realizar a través de la pasarela del Ethernet Shield W5100. Para ello, se ha de cargar la librería **Ethernet**.h, además de incluir en el código los siguientes fragmentos:

1. Se crea el entorno se conexión, asignando una mac al dispositivo, así como una IP, puerta de enlace (el router del hogar), con la máscara de red que proceda según el enmascaramiento asignado. Además, se crea la variable para el cliente con la llamada a la librería que se ha cargado en el punto anterior.

```
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress dnServer(192, 168, 1, 1);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress ip(192, 168, 1, 210);


EthernetClient ethClient;
```

2. Dentro de la función loop(), se lanza la conexión ethernet siempre y cuando esta no esté ya conectada. Para confirmar que Arduino se ha conectado a la red se

```
void loop() {
  if (!ethClient.connected()) {
    // initialize the ethernet device
  Ethernet.begin(mac, ip, dnServer, gateway, subnet);
  //print out the IP address
  Serial.print("IP = ");
  Serial.println(Ethernet.localIP());
```

ha establecido un punto de control en el que devuelve por Serial la IP asignada

4.2.3 Código Arduino para MQTT full dúplex

La función especial que se implementará en este proyecto en Arduino será el código que permita bidireccionalidad con el bróker MQTT, es decir, que Arduino tenga la capacidad tanto de suscribirse a un Topic determinado, como de Publicar información.

Para ello, han de diferenciarse diferentes funciones, además de la inicialización de librerías y variables para la conexión, con lo que el código ha de mantener una estructura básica:

1. Inicialización de librerías, entre las cuales ha de encontrarse **PubSubClient**.h

   La librería PubSubClient es un cliente que permite realizar operaciones simples de publicación/subscripción de cliente sobre un servidor MQTT.

2. Definición de la IP en la que se aloja el bróker MQTT, hacia el cual apuntará en subscripción y publicación de topic: (const char* mqtt_server="192.168.1.200";)
3. Establecimiento de la variable “cliente”, la cual se lanzará en la función Setup(). (**PubSubClient** mqttClient(ethClient);)
4. Definición de las diferentes funciones básicas para subscripción y publicación de Topics:
    a. void callback(char* topic, byte* payload, unsigned int length): Cuando se realiza la conexión con el bróker MQTT, desde la función Callback se reciben los mensajes entrantes en un Topic definido de suscripción. En esta función se procesa la respuesta del bróker, y se actuará con respecto a la información recibida.
    b. void reconnect(): En el caso de que se detecte un error de conexión, se lanzará esta función para reconectar tanto con las red local como con el servidor MQTT
    c. void mqttPublish(): En esta función se realiza la publicación de la información mediante los Topics definidos. Se añade además una variable temporal “millis”, que realiza la función de establecer un tiempo de espera entre publicaciones periódicas
        i. Dentro de otras funciones creadas en el programa también se realizará la publicación de información. Se diferencian de la función anterior en la periodicidad de la publicación, ya que estas últimas usarán un trigger para lanzar el mensaje.
5. Setup(): Configura la conexión del Arduino como cliente al bróker MQTT, además del callback que procesa los mensajes entrantes del bróker.
6. Loop(): A cada iteración sucede lo siguiente:
    i. El cliente MQTT se conecta (si no lo ha hecho ya).
    ii. Se subscribe a un topic determinado (o varios si así se estableciese) para comprobar si existen mensajes que procesar.
    iii. Se lanzan las diferentes funciones para leer información y publicarla en el Topic determinado si procede.

4.2.4 Uso de protocolo SPI

Una de las principales formas de comunicación disponibles en Arduino es mediante el bus SPI, el cual es un bus muy interesante ya que una gran variedad de sensores y dispositivos comerciales disponen de un interfaz SPI como medio de comunicación, por lo que se pueden controlar de manera más o menos sencilla desde Arduino.

En este proyecto se hará uso del bus SPI para tratar la información recibida por un lector de tarjetas RFID. Para ello en el código de programa ha de incluirse la librería <**SPI**.h>
El bus SPI tiene una arquitectura de tipo maestro-esclavo. El dispositivo maestro (master) puede iniciar la comunicación con uno o varios dispositivos esclavos (slave), y enviar o recibir datos de ellos. Los dispositivos esclavos no pueden iniciar la comunicación, ni intercambiar datos entre ellos directamente.
En el bus SPI la comunicación de datos entre maestros y esclavo se realiza en dos líneas independientes, una del maestro a los esclavos, y otra de los esclavos al maestro. Por tanto, la comunicación es Full Duplex, es decir, el maestro puede enviar y recibir datos simultáneamente.

Otra característica de SPI es que es bus síncrono. El dispositivo maestro proporciona una señal de reloj, que mantiene a todos los dispositivos sincronizados. Esto reduce la complejidad del sistema frente a los sistemas asíncronos.

Por tanto, el bus SPI requiere un mínimo de 3 líneas.

Figura 29 - Bus SPI

MOSI (Master-out, slave-in) para la comunicación del maestro al esclavo.
MISO (Master-in, slave-out) para comunicación del esclavo al maestro.
SCK (Clock) señal de reloj enviada por el maestro.
Su funcionamiento es sencillo, y es que el Master mantiene en High todas las lineas SS, y cuando quiere iniciar la comunicación con el Esclavo, pone en Low la línea SS correspondiente. En este proyecto no se utiliza todo el potencial del bus SPI, pero es de interés conocer la forma en el que el mismo actúa, puesto que se pueden añadir más funciones al sistema mediante este bus.

4.2.5 Código Habitación Principal:

Una vez vistas las partes de código más específicas, los 2 siguientes puntos se centran en el código utilizado para la maqueta del proyecto.

Para la habitación principal se usarán las siguientes librerías, seguidas de la configuración de red que se ha establecido en el diseño del proyecto:

```
#include <PubSubClient.h> → Esta librería dará capacidad a Arduino para interactuar con el servidor MQTT
#include <Ethernet.h> → Gracias a esta librería, Arduino podrá conectarse a la red local
#include <SimpleDHT.h> → Esta librería gestiona la información proporcionada por el sensor de temperatura y humedad
#include <DHT.h>  → Esta librería gestiona la información proporcionada por el sensor de temperatura y humedad


//Se realiza la configuración de red, asignando valores a las diferentes variables:

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress dnServer(192, 168, 1, 1);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress ip(192, 168, 1, 210);

const char* mqtt_server="192.168.1.200";

EthernetClient ethClient;
PubSubClient mqttClient(ethClient);
```

A continuación, se crean e inicializan las variables del programa, definiendo un nombre para los puertos de Arduino en cada caso:

```
#define ENCHUFE 6
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
#define LIGHT A8
#define RELELUZ 7
float hum=0, temp=0;
unsigned long previousMillis = 0 ; // Almacena el valor de tiempo en la última lectura
const long interval = 30000; // Intervalo entre lecturas del sensor DHT22 y luz
int normalizedValue;
int estadoluz;
const int DHTPin = 53;
DHT dht(DHTPin, DHTTYPE);
const int PIRPin= 49;
int estadoPIR=5;
```

Debido a que se necesitará conocer el estado real de la luz del dormitorio, debido a que al trabajar con un conmutador solo se dispone del estado real de la conmutación del relé, también se inicializan sus variables:

```
const int sensorIn = A0;
int mVperAmp = 185; // use 185 for 5A Module, 100 for 20A Module and 66 for 30A Module
double Voltage = 0;
double VRMS = 0;
double AmpsRMS = 0;
```

A partir de este punto se definen las diferentes funciones, en este caso serán:

- Leer mensajes MQTT con el topic subscrito, y lanzar las acciones dependiendo del código recibido:
    - void callback(char* topic, byte* payload, unsigned int length)
- Función para reconectar en caso de caída, tanto de la red local como del bróker MQTT
    - void reconnect()
- Función para lectura del DHT22 y del sensor de luz
    - void readDHT22()
- Función para publicar en MQTT los valores periódicos
    - void mqttPublish()
    - Se usará la función “millis” para establecer un tiempo entre medidas, con objeto de no saturar la red con medidas continuadas. Esto es por 2 motivos: El primero, que las lecturas del sensor DHT22 son lentas, en torno a los 2 segundos, por lo que si las lecturas son continuadas la placa Arduino estará prácticamente todo el tiempo saturado. Y 2, porque la información que recibiría Openhab entre lecturas sería prácticamente el mismo, por lo que se establece un tiempo entre medidas de 30 segundos. Esta medida está indicada en el código inicial:
        - const long interval = 30000
- Función para establecer nivel de corriente en el sensor ACS712, lee la función getVPP()
    - void Luz()
- Función para leer el sensor ACS712
    - float getVPP()
- Función para leer el sensor de movimiento
    - void Pir()
- Función Setup(), en la que se inicializan los valores de conexión y el estado de los pines (entrada o salida)

```
void setup() {
  Serial.begin(9600);
  mqttClient.setServer(mqtt_server, 1883);
  mqttClient.setCallback(callback);

  pinMode(ENCHUFE, OUTPUT);
  pinMode(RELELUZ, OUTPUT);
  pinMode(PIRPin, INPUT);

}
```

- Función Loop(), en la que se lanzan las funciones definidas para el correcto funcionamiento del sistema:

```
void loop() {
  if (!mqttClient.connected()) {
    // initialize the ethernet device
  Ethernet.begin(mac, ip, dnServer, gateway, subnet);
  //print out the IP address
  Serial.print("IP = ");
  Serial.println(Ethernet.localIP());
    reconnect(); //Función para reconectar en caso de caída
  mqttClient.subscribe("myhome/bedroom/#");

  }
  //Lanzamos las diferentes funciones dependientes
  Luz();
  Pir();
  mqttPublish(); // Publica los datos con el protocolo MQTT
 delay (300) ;

   mqttClient.loop();


}
```

4.2.6 Código Simulación

El sketch del Arduino que se ha usado para la Simulación de la maqueta se mantiene el mismo esquema, pero con algunas diferencias.

El código de Simulación incluye la lectura de sensores para simular la información recibida en el hogar, así como diferentes alarmas (fuego, gas, inundación), y hasta la parte de código para activar o desactivar la alarma mediante una tarjeta RFID, por lo que sus funciones serán:

```
#include <PubSubClient.h>
#include <SPI.h>
#include <Ethernet.h>
#include <SimpleDHT.h>
#include <DHT.h>
#include <MFRC522.h>

byte mac[] = {0xDE, 0xDA, 0xBE, 0xEB, 0xFE, 0xED};
IPAddress dnServer(192, 168, 1, 1);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress ip(192, 168, 1, 225);

const char* mqtt_server="192.168.1.200";

//****************************Parámetros de RFID*************************************************
#define SS_PIN 53
#define RST_PIN 8

MFRC522 rfid(SS_PIN, RST_PIN);

MFRC522::MIFARE_Key key;
// Init array that will store new NUID
byte nuidPICC[3];
#define NEW_UID {0xDE, 0xAD, 0xBE, 0xEF}
String UIDstring;
```

A continuación, se definen los sensores y actuadores, así como las funciones Callback(), MqttPublish(), readDHT22(), o reconnect(), y se añaden las funciones necesarias para la lectura de RFID:

- void leer_rfid()
- void printHex(byte *buffer, byte bufferSize)
- void printDec(byte *buffer, byte bufferSize)

Además de las diferentes funciones para las alarmas de gas y fuego, o de inundación:

- void alarmas()
- void agua()

Finalmente, en la función setup() se inicializan los valores de conexión y el estado de los pines (entrada o salida), además de inicializar el bus SPI:

```
//**********************RFID**********************
SPI.begin();          // Init SPI bus
  rfid.PCD_Init();      // Init MFRC522

  for (byte i = 0; i < 6; i++)
    {  key.keyByte[i] = 0xFF;
    }

  Serial.println(F("This code scan the MIFARE Classsic NUID."));
  Serial.print(F("Using the following key:"));
  printHex(key.keyByte, MFRC522::MF_KEY_SIZE);
//*************************************************************
```

Y se configura la función Loop(), en la que se lanzan las funciones definidas para el correcto funcionamiento del sistema:

```
void loop() {
  if (!mqttClient.connected()) {
    // initialize the ethernet device
  Ethernet.begin(mac, ip, dnServer, gateway, subnet);
  //print out the IP address
  Serial.print("IP = ");
  Serial.println(Ethernet.localIP());
    reconnect();
  mqttClient.subscribe("myhome/#");

  }
if (estadoalarma==1){
  digitalWrite(ALARM_ON, HIGH);
  delay(500);
    digitalWrite(ALARM_ON, LOW);
}
  mqttPublish(); // Publica los datos con el protocolo MQTT
 alarmas();
 agua();
 leer_rfid();

 delay (300) ;

   mqttClient.loop();

}
```

4.3 Pruebas E/S

Una vez cargada la configuración en los Arduino se comprueba si se pueden leer los mensajes MQTT sobre la red:

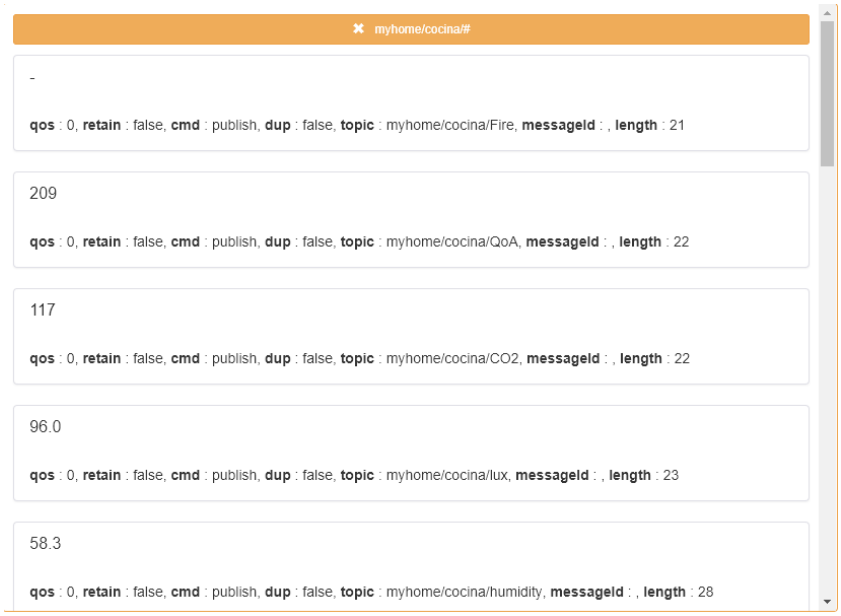
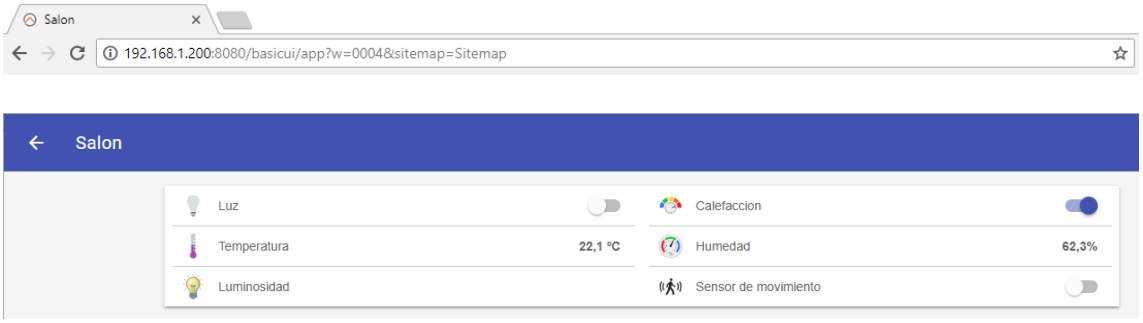


Figura 30 - Pruebas Sistema

El bróker MQTT está recibiendo los mensajes procesados por Arduino. De la misma forma, para comprobar que el Arduino lee los mensajes proporcionados por Openhab, se encenderá la calefacción del salón:

Por último, se comprueba que la información que se publica en el servidor MQTT está siendo leída por Openhab, por lo que se accede vía Web:



Se confirma que existe conectividad entre ambos dispositivos, vía protocolo MQTT como se había propuesto. Se realizarán el resto de las pruebas en video para demostrar el completo funcionamiento del sistema.

4.4 Configuración de reglas

Para finalizar, se establecerán reglas por las cuales en el momento en el que la temperatura de una habitación baje de una temperatura seleccionada, se encienda la calefacción.

Se realiza la siguiente configuración en el fichero Heat.rules:

```
1   rule "Salon"
2   when
3       Item GF_LivingRoom_Temperature received update or
4       Item GF_LivingRoom_Thermostat received update
5   then
6       if (GF_LivingRoom_Temperature.state < GF_LivingRoom_Thermostat.state) {
7           GF_LivingRoom_Heating.sendCommand(ON)
8       }
9       if (GF_LivingRoom_Temperature.state > GF_LivingRoom_Thermostat.state) {
10          GF_LivingRoom_Heating.sendCommand(OFF)
11      }
12  end
```

La lógica de las reglas es la siguiente:

- rule: Nombre de la regla
- when: Se establece cuando se ejecutará la regla, cuál será el trigger de la misma. En este caso la regla se ejecutará cada vez que haya un cambio de temperatura en el salón, o que desde la aplicación se cambie la configuración del termostato

- then: acciones a ejecutar. En este caso, si la temperatura actual es menor que el valor indicado en el termostato, se encenderá la calefacción del salón. En caso contrario se apagará.

Se observa que el sistema es completamente funcional, la cual permite ahorrar costes al tener un total control sobre los consumos del hogar.

Además, se han incluido los siguientes ficheros con reglas adicionales, cuyo funcionamiento se verá durante la presentación:

- Alarm: Lee los valores de la tarjeta RFID utilizada para activar o desactivar la alarma. En caso de ser una tarjeta conocida ejecuta la acción.
- Arena: Se ha establecido un proceso-recordatorio sin ningún uso de sensores. En este caso, se ha incluido un sistema en el cual 2 veces al día se recibe un recordatorio en nuestro terminal para limpiar la arena de nuestra mascota. Una vez limpia, se marca la arena como “limpia”, y en el momento del recordatorio se marca automáticamente como “sucia”. Este sistema se implementa para demostrar que las aplicaciones del sistema son infinitas, y no necesariamente es necesario HW para ejecutarlas.
- Iphone: Se han creado unas reglas que utilizan los bindings del sistema, en este caso el Binding de Icloud, para crear un interface del sistema con nuestro teléfono. Con esta regla se dispone incluso de un mapa con la localización, y un icono que muestra si el terminal se encuentra o no en el hogar.
- Motion: Esta regla se utiliza para marcar la hora de la última detección del sensor de movimiento.

4.5 Implementación de gráficas en Grafana

Uno de los elementos adicionales del sistema será la implementación de gráficas por medio de la aplicación Grafana, la cual ha de estar instalada y configurada como servidor dentro de la Raspberry Pi, además de encontrarse enlazada con la BBDD InfluxDB.

Para implementar una gráfica en el sistema se ha de crear la gráfica dentro de la aplicación de Grafana, que en este caso se encuentra accesible desde el puerto 3000 como se había diseñado y especificado, y darle visibilidad mediante una página web local, a la cual tendrá acceso Openhab.

Para lograr esto, se crea un fichero html con el siguiente texto, según el enlace que crea la propia aplicación:

<iframe src="http://192.168.1.200:3000/d-solo/MzVBKskRk/salon?orgId=1&panelId=2&from=now-24h&to=now&theme=light" width="420" height="200" frameborder="0"></iframe>

Una vez creado, se guarda el fichero en la carpeta de configuración de Openhab, dentro de la subcarpeta “html”, y se crea un enlace en el Sitemap con el siguiente texto:

```
Frame label="Hab Principal Grafana" {
Webview url="http://192.168.1.200:8080/static/grafana.html" height=6
}
```

De esta forma, se tiene acceso al gráfico generado con Grafana, más amigable que los creados de forma nativa por Openhab.

Gráfico Openhab:

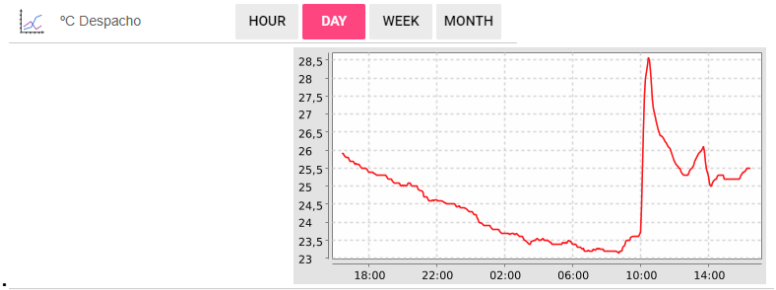


Gráfico Grafana:

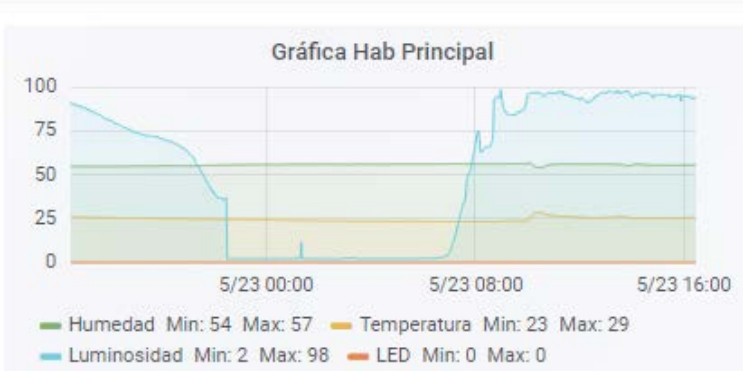


Figura 31 - Gráficas Sistema

4.6 Streaming de Video IP

Se creará un proceso en Python con el código que se puede obtener de la web de Raspberry Pi [18], y según procedimiento de un paso a paso localizado en la web [19].

Este proceso creará un Streaming que podrá ser compartido en Openhab a través de la dirección 192.168.1.200:8000, y el cual se enlazará en el Sitemap de proyecto para poder acceder al video en directo.

```
Frame label="RaspiCam" {
                    Text label="RaspiCam" icon="video" {
          Webview url="http://192.168.1.200:8000/index.html" height=15
          }
  }
```

Figura 32 - Video Camara IP

4.7 Creación de la Maqueta

Para disponer de un medio de pruebas y comprobaciones del sistema, se desarrolla una maqueta que incluye por un lado la planta baja del hogar impresa en plástico (Se adjuntan los STLs con la documentación), y por otra un diseño de la habitación principal de la planta superior, en la cual se han instalado conmutadores mecánicos para el control de luz.

Toda la maqueta se consolida en un tablón en el que se diferencian varias partes:

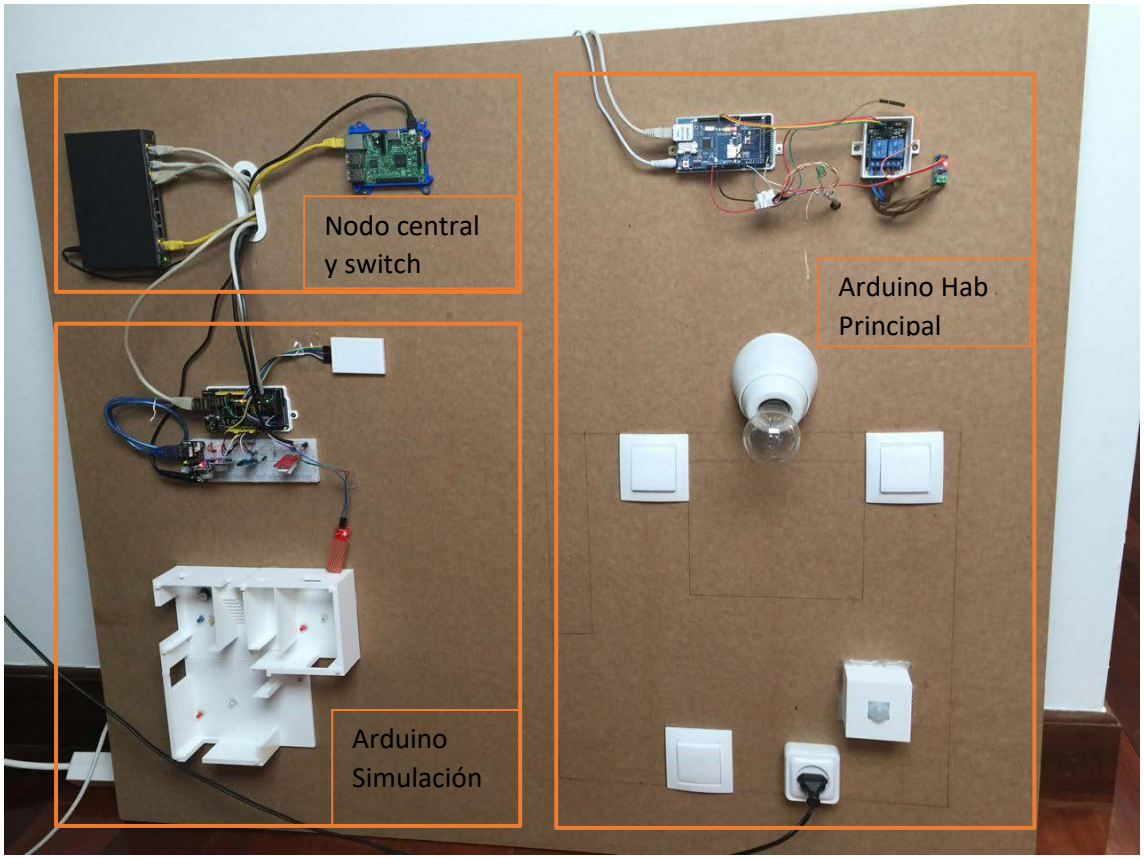


Figura 33 - Maqueta

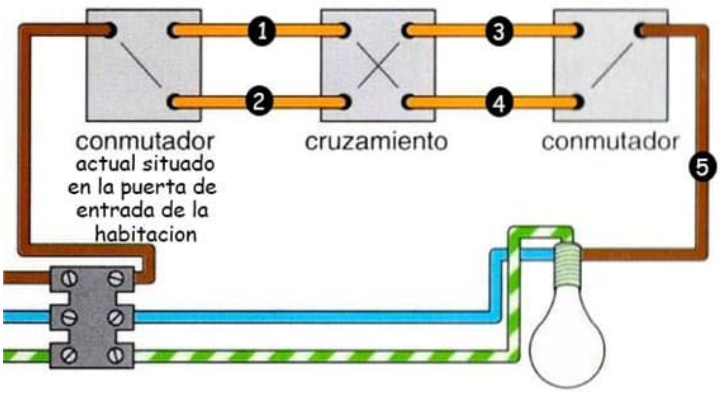
4.5.1 Diseño maqueta planta baja

Se ha realizado impreso la maqueta en plástico PLA (ácido poliláctico) mediante el diseño y la impresión mediante impresora 3D. Se han incluido las diferentes estancias de la planta baja con varios leds para simular luces y calefacción. Además, sobre protoboard, se han incluido sensores indicados en la propuesta destinados a la cocina.

4.5.2 Diseño maqueta habitación principal

Para el diseño de la habitación principal se ha tenido en cuenta probablemente el apartado más complicado de gestionar de esta parte del proyecto, la gestión de luz.

Esto es debido a que se han instalado conmutadores mecánicos, de los cuales se desconoce el estado en cada momento. El esquema sería el siguiente:



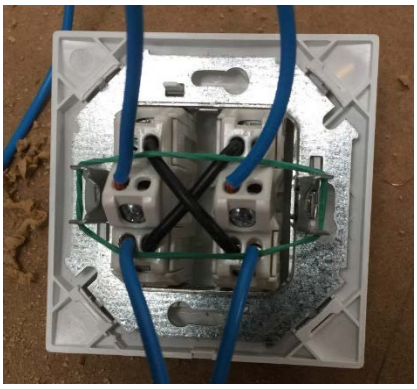


Figura 34 - Esquema Electrico Luz

En este caso, uno de los conmutadores de la maqueta (1 extremo) será el propio relé, luego se instalarán 2 cruzamientos, y finalmente se instalará el 2º conmutador.

De esta forma, una vez realizadas las pruebas correspondientes, se observa el correcto funcionamiento del sistema pulsando el switch en openhab para encender o apagar la luz, ya que el relé funciona como conmutador.

En este punto existe un problema. El sistema conoce la posición del relé, pero no la de los conmutadores del sistema, por lo que se podría estar observando en Openhab que el relé está encendido, pero la luz apagada, y viceversa. Para solucionar este problema se hace uso de un sensor de corriente ACS712 conectado al relé que da servicio a la

```
// ----------------Función para establecer nivel de corriente en el sensor ACS712, lee la función getVPP()------

void Luz(){
 Voltage = getVPP();
 VRMS = (Voltage/2.0) *0.707;
 AmpsRMS = (VRMS * 1000)/mVperAmp;
 Serial.print(AmpsRMS);
 Serial.println(" Amps RMS");


if (AmpsRMS>0.15 and estadoluz==0){
  estadoluz=1;
   mqttClient.publish("myhome/bedroom/luz", "1");
  }
 if (AmpsRMS<0.15 and estadoluz==1){
  estadoluz=0;
  mqttClient.publish("myhome/bedroom/luz", "0");
  }
  }
```

bombilla, y a través del cual se conocerá el estado de la luz, no el del relé, implementando el siguiente código:

```
// -----------------Función para leer el sensor ACS712-------------

float getVPP(){
  float result;

  int readValue;             //value read from the sensor
  int maxValue = 0;          // store max value here
  int minValue = 1024;          // store min value here

   uint32_t start_time = millis();
   while((millis()-start_time) < 1000) //sample for 1 Sec
   {
       readValue = analogRead(sensorIn);
       // see if you have a new maxValue
       if (readValue > maxValue)
       {
           /*record the maximum sensor value*/
           maxValue = readValue;
       }
       if (readValue < minValue)
       {
           /*record the maximum sensor value*/
           minValue = readValue;
       }
   }

   // Subtract min from max
   result = ((maxValue - minValue) * 5.0)/1024.0;

   return result;
 }
```

Para ello, además, se ha de crear el siguiente enlace MQTT en Openhab, mediante el cual se escribe sobre el ítem switch, pero se lee el estado del ACS712, todo ello con el mismo ítem:

```
>[broker:myhome/bedroom/rele:command:ON:1]
>[broker:myhome/bedroom/rele:command:OFF:0]
<[broker:myhome/bedroom/luz:state:ON:1]
<[broker:myhome/bedroom/luz:state:OFF:0]
```

De esta forma, se controla de forma manual el estado del switch, pero este mismo trata siempre se conocer su posición a través de la información del sensor. De esta forma, el sistema es completamente funcional.

Por último, y con objeto de lograr un diseño del sistema acorde con las instalaciones actuales, se ha diseñado un pequeño prototipo para encastrar en cualquier instalación habitual, ya que mantiene las medidas típicas de 7,4x7,4cm:

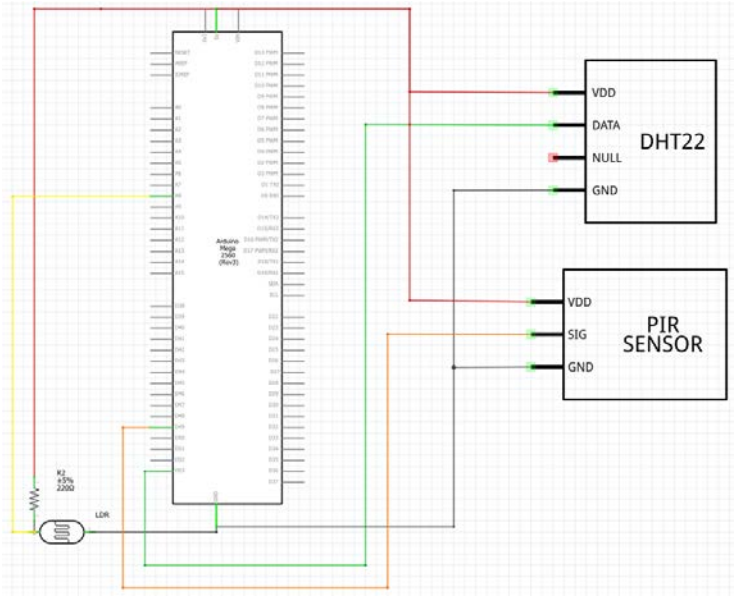
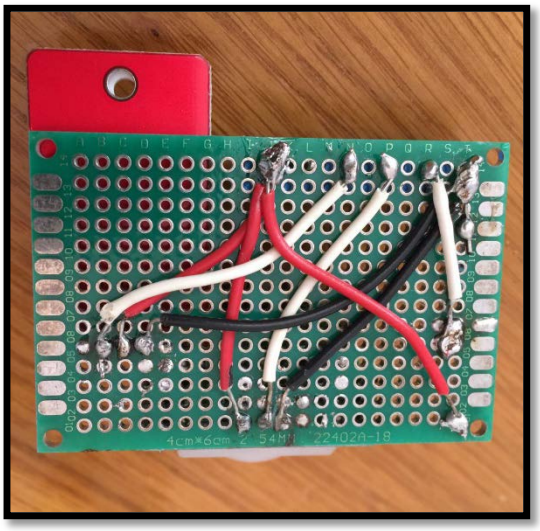
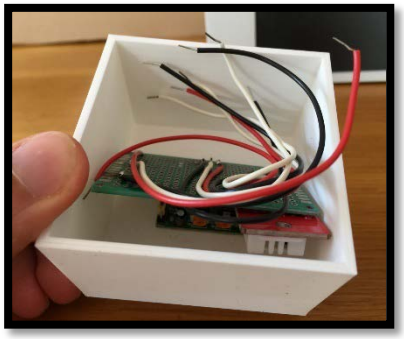


Figura 35 - Esquema Prototipo



Con la instalación de este prototipo se finaliza la maqueta, la cual queda preparada para la fase de pruebas que se realizará en video.



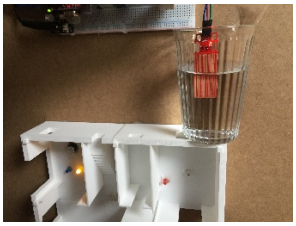
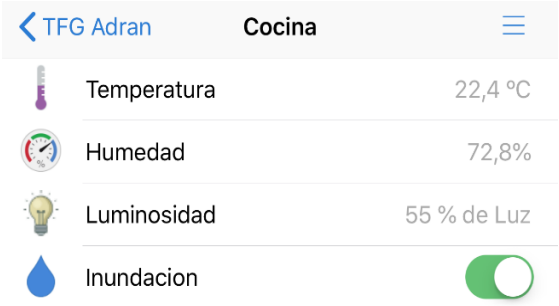
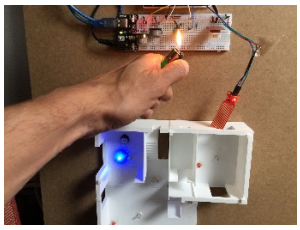
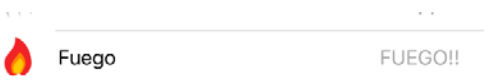
CAPÍTULO 5 – PRUEBAS Y RESULTADOS

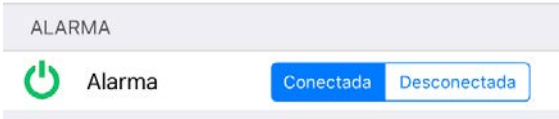
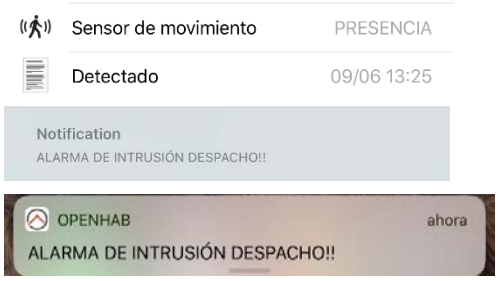
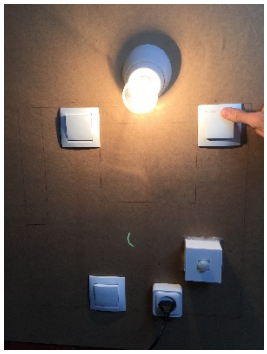
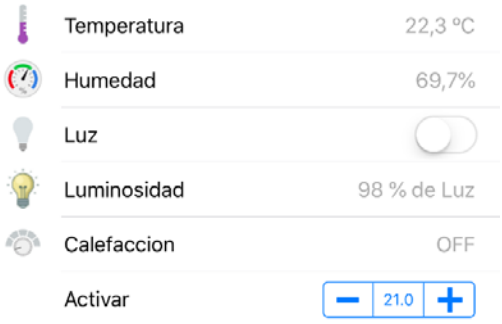
5.1 Pruebas del sistema

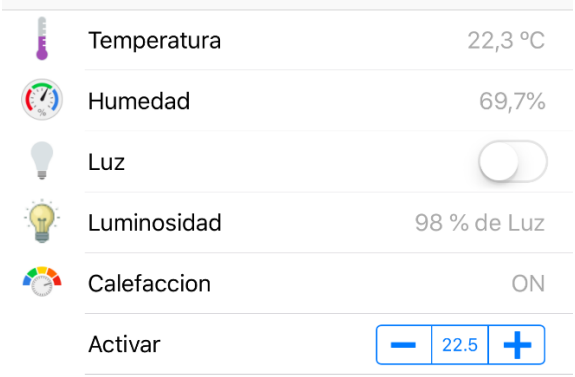
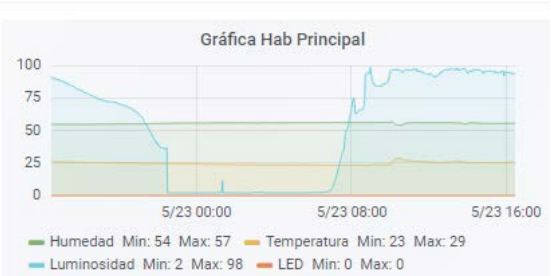
Una vez finalizado el trabajo de proyecto, se establecen una serie de pruebas de control para verificar que todos los sistemas funcionan correctamente. Estas pruebas se entregarán además en un video de demostración para mayor visibilidad de la solución.

Las pruebas básicas serán:

- Alarmas: Se hacen saltar las alarmas fuego e inundación
- Presencia: Se activa la alarma mediante RFID y se comprueba su activación en maqueta (led indicador), y se hace saltar la alarma para recibir notificación en el terminal móvil.
- Gestión de Luz: Se encenderá y apagará la luz desde la aplicación o desde los mecanismos mecánicos, conociendo en cada instante el estado real de la bombilla
- Calefacción: Se activa la calefacción marcando una Tª objetivo superior a la actual de la estancia
- Streaming: Comprobación de funcionamiento del video desde la aplicación
- Grafana: Comprobación de muestra de gráficas de Grafana en la aplicación

Prueba	Tipo	Foto Maqueta (si procede)	Aplicación (BasicUI)
ALARMAS	Inundación		
	Fuego		

PRESENCIA	Activar RFID		
	Alarma de presencia		
GESTIÓN DE LUZ	Encender desde mecanismo		
	Apagar desde BasicUI		
CALEFACCIÓN	Desactivada		

	Activada		
STREAMING	Video		
GRAFANA	Gráfica		

Además, se mostrarán otra serie de funciones que se han incluido en la aplicación, las cuales aportan valor añadido al sistema, como el control del televisor del hogar, obtener información del ciclo solar y lunar, estado del sistema, gráficas...

5.2 Resultados

Una vez finalizado el proyecto, y como último punto, se revisan los requisitos establecidos en una primera fase de proyecto para poder concluir con éxito el proyecto de fin de grado:

Nº	Descripción	Resultado
1	Conectividad E2E de la solución usando el protocolo M2M MQTT	OK
2	Uso de BBDD InfluxDB para servicios de persistencia e interconexión con Grafana	OK
3	Gestión de servicios de streaming de video IP con la cámara nativa de Raspberry PI	OK
4	Gestión de encendido y apagado de luces mediante plataforma Openhab, conociendo en cada momento	OK
5	Activación y desactivación de la alarma del hogar por medio de una tarjeta RFID	OK
6	Gestión de la calefacción del hogar, de forma personalizada y dependiente de la temperatura de cada estancia	OK
7	Gestión de alarmas en caso de escape de gases, fuego o inundación	OK
8	Solución completa con servicios integralmente Open Source	OK
9	Solución completa con presupuesto <7000€ (Iva incluido)	OK

Tabla 6 - Resultados

Se observa que se han logrado los objetivos de proyecto, ya que se han cumplido de forma objetiva los requisitos marcados en el inicio de esta memoria, por lo que se valora como un éxito vistos los resultados que aporta.

Aunque se entiende que el proyecto ha sido un éxito, también se aprecian multitud de posibilidades de mejora. Una de ellas, y la más importante, sería la de adaptar un ESP8866 como nodo remoto, para incluir de forma sencilla, e inalámbrica, más funcionalidades al sistema de forma económica. Esto permitiría, por ejemplo, monitorizar un objeto en el hogar que no siempre se encuentre fijo en un lugar determinado, o que, por el motivo que sea, no tenga próxima una conexión Ethernet.

Figura 36 - ESP8266

Se concluye el proyecto con la reflexión de la cantidad de facilidades que aporta una solución como la propuesta para multitud de escenarios, tal y como son un hogar, una residencia de ancianos, un hospital… y en la cantidad de aplicaciones adicionales que un sistema Open Source permite implementar, haciendo que la parte de ingeniería de la solución sea lo importante en este tipo de proyectos, no siendo necesaria la aportación de material o software propietario para conseguirlo.

BIBLIOGRAFÍA

[1] Raspbian – “Software”, https://www.raspberrypi.org/downloads/raspbian/

[Último acceso: 15 de marzo de 2018]

[2] VNC – “Software”, https://www.realvnc.com/es/connect/download/viewer/ [Último acceso: 16 de marzo de 2018]

[3] Putty – “Software”, https://www.putty.org/ [Último acceso: 16 de marzo de 2018]

[4] Win32 Disk imager – “Software”, https://sourceforge.net/projects/win32diskimager/ [Último acceso: 17 de marzo de 2018]

[5] Openhab, https://docs.openhab.org/introduction.html [Último acceso: 11 de abril de 2018]

[10] Instalación Mosquitto, https://mosquitto.org/blog/2013/01/mosquitto-debian-repository/ [Último acceso: 17 de marzo de 2018]

[11] Instalación Mosquitto, https://programarfacil.com/esp8266/mqtt-esp8266-raspberry-pi/#Instalacion_Mosquitto_en_Raspberry_Pi [Último acceso: 17 de marzo de 2018]

[12] Openhab Docs – Instalación Samba, https://docs.openhab.org/installation/linux.html [Último acceso: 20 de marzo de 2018]

[13] Notepad++ – “Software”, https://notepad-plus-plus.org/download/v7.5.5.html [Último acceso: 24 de marzo de 2018]

[14] Openhab Docs – Configuracion https://docs.openhab.org/configuration/index.html [Último acceso: 8 de abril de 2018]

[15] Nociones - Instalación InfluxDB, https://www.nociones.de/instalacion-y-primeros-pasos-con-influxdb-en-ubuntu-18-04/ [Último acceso: 3 de abril de 2018]

[16] Tech.Scargill - Instalación InfluxDB + Grafana, https://tech.scargill.net/grafana-and-influxdb/ [Último acceso: 3 de abril de 2018]

[17] down – Instalación InfluxDB + Grafana http://www.d0wn.com/install-grafana-4-on-a-pi/ [Último acceso: 4 de abril de 2018]

[18] Raspberry Pi Camara - http://picamera.readthedocs.io/en/latest/recipes2.html [Último acceso: 4 de Mayo de 2018]

[19] Random Nerd Tutorials - Streaming IP - https://randomnerdtutorials.com/video-streaming-with-raspberry-pi-camera/ [Último acceso: 4 de Mayo de 2018]

[20] Wikipedia – "Domótica", https://es.wikipedia.org/wiki/Dom%C3%B3tica [Último acceso: 27 de Abril de 2018]

[21] Wikipedia – "Raspberry Pi", https://es.wikipedia.org/wiki/Raspberry_Pi [Último acceso: 27 de Abril de 2018]

[22] CEDOM, Asociación española de Domótica e Inmótica, http://www.cedom.es/es [Último acceso: 30 de Abril de 2018]

[23] Domótica para todos, http://domoticaparatodos.com/2016/11/pros-y-contras-de-z-wave/ [Último acceso: 29 de Abril de 2018]

[24] Hogartec, http://hogartec.es/hogartec2/sistemas-domoticos-centralizados-descentralizados-y-distribuidos/ [Último acceso: 29 de Abril de 2018]

[25] Geeky Theory, https://geekytheory.com/que-es-mqtt [Último acceso: 10 de Abril de 2018]

[26] Luis Llamas, https://www.luisllamas.es/medir-nivel-luz-con-arduino-y-fotoresistencia-ldr/ & https://www.luisllamas.es/arduino-salida-rele/ [Último acceso: 27 de Abril de 2018]

[27] IBM Knowledge Center, https://www.ibm.com/support/knowledgecenter/es/SS9D84_1.0.0/com.ibm.mm.tc.doc/tc00000_.htm [Último acceso: 10 de Mayo de 2018]

ANEXO I – Instalación Básica

I. Instalación de software en Raspberry Pi

En este apartado se describen los diferentes pasos para realizar la instalación y la configuración del sistema operativo completo en la Raspberry Pi 3. Se divide en los siguientes apartados:

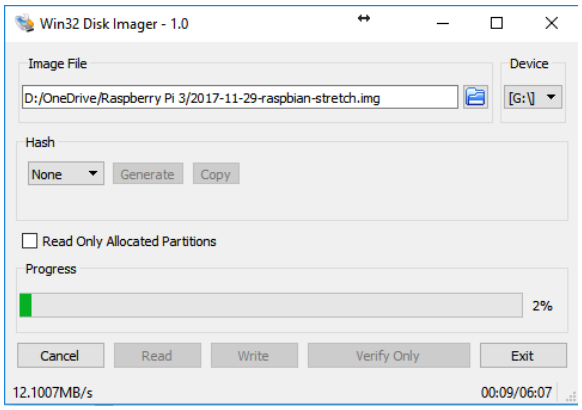
- Instalación Raspbian
- Configuración básica Raspbian para OpenHAB
- Instalación Openhab
- Configuración básica OpenHAB

Instalación Raspbian

Raspbian [1] es el sistema operativo por excelencia de cualquier modelo de Raspberry Pi, aunque no es el único que se adapta a la misma. En este trabajo se realizará la instalación y configuración básica de dicho sistema operativo por la gran compatibilidad con el sistema en conjunto. Esta configuración se realizará en modo escritorio -por lo que se descargará la versión Desktop de Raspbian- conectando la Raspberry Pi a un monitor y a un teclado y ratón. A partir de este punto no será necesario nada más que conectar la Raspberry a una conexión de área local, puesto que se habilitará la conexión SSH para realizar los siguientes pasos en remoto.

Una vez descargado el sistema operativo de la propia web de Rasberry (https://www.raspberrypi.org/downloads/raspbian/) se realizará la instalación en una tarjeta SD de al menos 16Gb. Se hará uso del SW “Win32 Disk imager” para cargar en la tarjeta SD el sistema operativo:

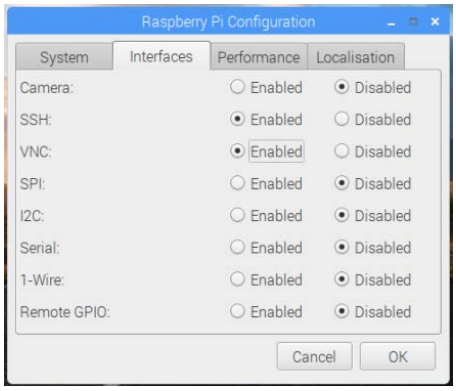
1. Cargar la imagen desde la dirección en la que esté ubicada
2. Seleccionar la letra de la unidad en la que se encuentre la SD, puesto que si no podría borrarse información de otra unidad
3. Pulsar sobre Write y esperar:



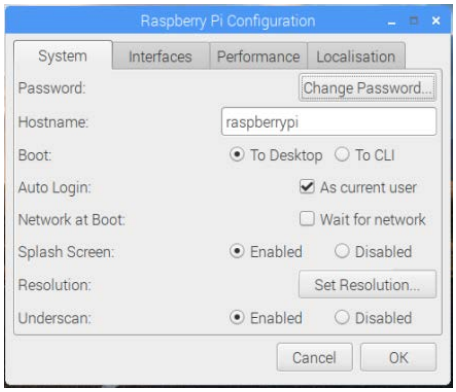
Una vez finalizada la carga de la imagen en la SD, se debería de recibir el mensaje de "Write Successful". Se introduce la tarjeta en la ranura de la Raspberry y se enciende el equipo.

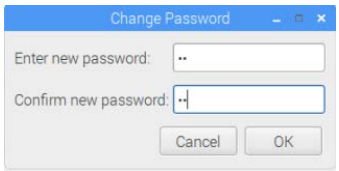
Configuración básica Raspbian para OpenHAB

Como se ha comentado al comienzo del apartado, se hará uso de un monitor mediante una conexión HDMI y un ratón y teclado USB para la configuración básica que permita habilitar la conexión por SSH y VNC.

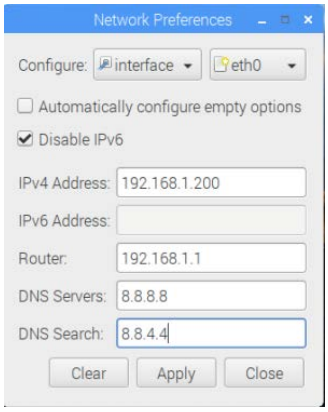


Y se modifica el pass del usuario por defecto de la Raspberry.





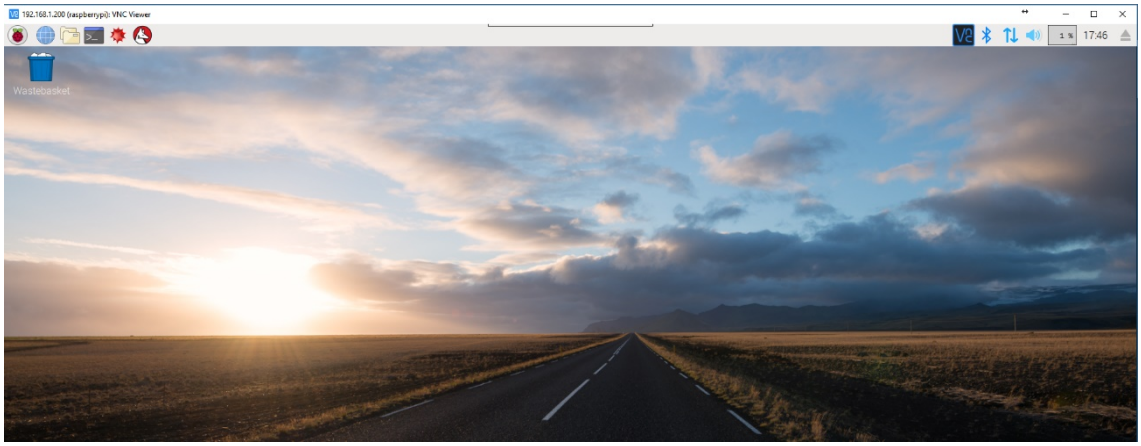
Por último, se realiza la configuración IP para establecer un direccionamiento estático y alejado de los rangos habituales de la red local. En este caso se configura el que se había establecido en el diseño de la solución:



Con esta configuración, la Raspberry Pi es completamente autónoma en el sentido que ya no es necesario ni un monitor ni ratón ni teclado para su configuración, únicamente necesita una conexión de red permanente. Para comprobar que todo ha sido cargado con éxito, se prueban ambos métodos de conexión dejando la Raspberry simplemente conectada a la red de área local:

Vía VNC:

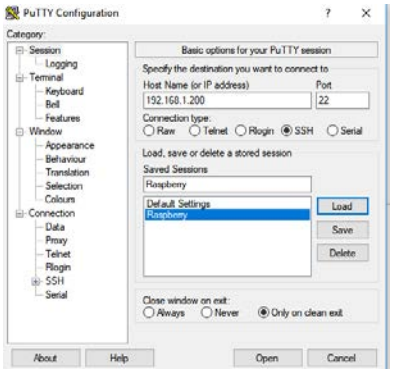
Se instala VNC [2] en un PC y se apunta a la dirección de la Raspberry (192.168.1.200), con el usuario y password definido.



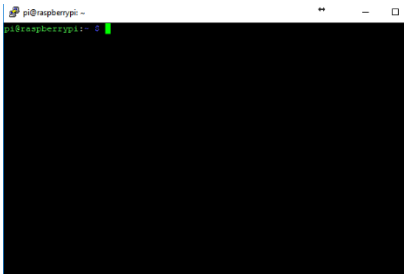
Vía SSH:

Se instala el software Putty [3] si se trabaja con Windows, o se abre una consola de Linux para establecer una conexión SSH.

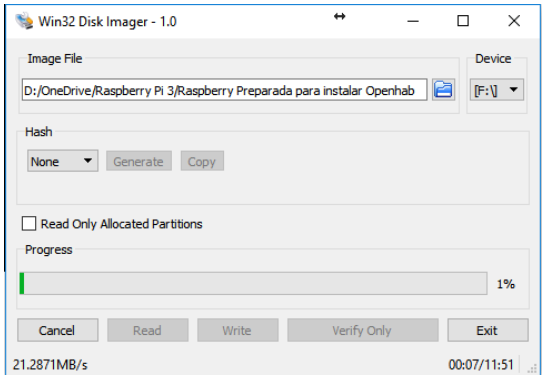
Se apunta a la Raspberry de nuevo con SSH 192.168.1.200:22 desde Linux, o simplemente se marca la opción SSH en Putty y se direcciona la misma IP:



Para acceder, con el mismo user y pw a la Raspberry:



En este punto se realizará una copia de seguridad del estado de la tarjeta SD con Win32 disk Imager [4], puesto que en caso de problemas se podría retomar la configuración desde este punto:



Se establece un nombre a la copia de seguridad en una ubicación elegida, se selecciona la partición de la ubicación de la SD, y se pulsa READ.

ANEXO II – Instalación OpenHab

I. Instalación Openhab

Para realizar la instalación de Openhab [5], se tendrá en cuenta la guía de configuración que proporciona la propia plataforma. (https://docs.openhab.org/installation/index.html)

- **Preconfiguración:**

Para comenzar, se realiza la conexión por SSH, y se ejecuta el comando sudo raspi-config para ajustar el sistema a la instalación de OpenHAB:

- Se expande el sistema de ficheros
- Se puede cambiar el hostname (en este caso no se ha hecho)
- Se modifica la asignación de memoria a la GPU a 16Mb, y el resto de la memoria se asignará a la CPU, debido a que el uso que la Raspberry haga de la memoria gráfica será prácticamente residual, y se potencia la CPU.

Una vez hecho esto, se reinicia el sistema para que actualice los cambios, y se realiza una carga completa de actualizaciones de paquetes en la Raspberry:

```
sudo apt-get update
sudo apt-get upgrade
```

Una vez finalizada la carga se procede con su configuración:

Uno de los requisitos de Openhab es disponer de Java instalado en el equipo. En este punto Openhab propone 3 alternativas: Zulu, Java o OEP…

Se opta por Zulu por disponer de licencia gratuita Open Source, lo cual va en línea con este trabajo:

Primero se instala el paquete dirmngr:

```
sudo apt-get install dirmngr
```

Y a continuación se cargan los siguientes comandos:

```
sudo su
apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 0x219BD9C9
```

```
echo 'deb http://repos.azulsystems.com/debian stable main' > /etc/apt/sources.list.d/zulu.list
apt-get update -qq
apt-get install zulu-embedded-8
```

Y se verifica la versión de Java:

```
java -version
```

Asegurándose que se está usando Zulu como predeterminado:

```
update-alternatives --config java
```

```
root@raspberrypi:/home/pi# java -version
openjdk version "1.8.0_152"
OpenJDK Runtime Environment (Zulu Embedded 8.25.0.76-linux-aarch32hf) (build 1.8
.0_152-b76)
OpenJDK Client VM (Zulu Embedded 8.25.0.76-linux-aarch32hf) (build 25.152-b76, m
ixed mode, Evaluation)
root@raspberrypi:/home/pi# update-alternatives --config java
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                            Priority
 Status
------------------------------------------------------------
* 0            /usr/lib/jvm/zulu-embedded-8-armhf/bin/java      1082500
 auto mode
  1            /usr/lib/jvm/jdk-8-oracle-arm32-vfp-hflt/jre/bin/java   318
 manual mode
  2            /usr/lib/jvm/zulu-embedded-8-armhf/bin/java      1082500
 manual mode

Press <enter> to keep the current choice[*], or type selection number: 0
root@raspberrypi:/home/pi#
```

- **Instalación:**

Lo primero que se tiene que hacer para instalar el sistema Openhab, es añadir el repositorio de Bintray al gestor de paquetes y permitir al Apt usar el protocolo HTTPS:

```
wget -qO - 'https://bintray.com/user/downloadSubjectPublicKey?username=openhab' | sudo apt-key add -
sudo apt-get install apt-transport-https
```

En este punto se instalará la versión de Openhab que se decida. Esta puede ser la versión Estable, la beta o la de Testing (de más a menos estable). Se selecciona la versión estable, que es la que más garantías ofrece para el funcionamiento del sistema:

Se añade el repositorio de la versión Estable a la lista de Apt:

```
echo 'deb https://dl.bintray.com/openhab/apt-repo2 stable main' | sudo tee
/etc/apt/sources.list.d/openhab2.list
```

Se resincroniza el índice de paquetes:

```
sudo apt-get update
```

Y se procede a lanzar la instalación de Openhab con el siguiente comando:

```
sudo apt-get install openhab2
```

```
pi@raspberrypi:~ $ sudo apt-get install openhab2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  openhab2
0 upgraded, 1 newly installed, 0 to remove and 1 not upgraded.
Need to get 68.6 MB of archives.
After this operation, 76.6 MB of additional disk space will be used.
Get:1 https://dl.bintray.com/openhab/apt-repo2 stable/main armhf openhab2 all 2.
2.0-1 [68.6 MB]
Fetched 68.6 MB in 18s (3,658 kB/s)
Selecting previously unselected package openhab2.
(Reading database ... 124901 files and directories currently installed.)
Preparing to unpack .../openhab2_2.2.0-1_all.deb ...
Unpacking openhab2 (2.2.0-1) ...
Setting up openhab2 (2.2.0-1) ...
### NOT starting on installation, please execute the following statements to con
figure openHAB to start automatically using systemd
 sudo /bin/systemctl daemon-reload
 sudo /bin/systemctl enable openhab2.service
### You can start openhab2 by executing
 sudo /bin/systemctl start openhab2.service
Processing triggers for systemd (232-25+deb9u2) ...
```

Como punto adicional, también se instalarán los Add-on offline por si en algún momento existiese algún problema con la conexión, ya que así seguiría funcionando la máquina en red local:

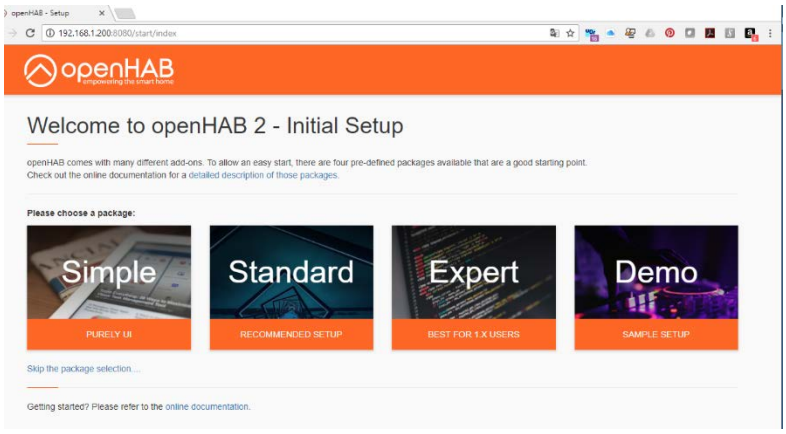
```
sudo apt-get install openhab2-addons
```

Una vez en este punto, solo queda lanzar la aplicación y configurarla para que se inicie automáticamente cada vez que se reinicie el sistema:

```
sudo systemctl start openhab2.service
sudo systemctl status openhab2.service

sudo systemctl daemon-reload
sudo systemctl enable openhab2.service
```

Ahora, se abre una página en un explorador y se lanza 192.168.1.200:8080



Ya se dispone de OpenHAB correctamente instalado en el núcleo del sistema, la Raspberry. Se realiza una última prueba, lanzar el comando *sudo reboot*, para

comprobar que la aplicación se lanza automáticamente si se reiniciase el sistema. Se obtiene un resultado positivo, con la misma pantalla que la imagen anterior, por lo que se realiza una nueva copia de seguridad en este punto.

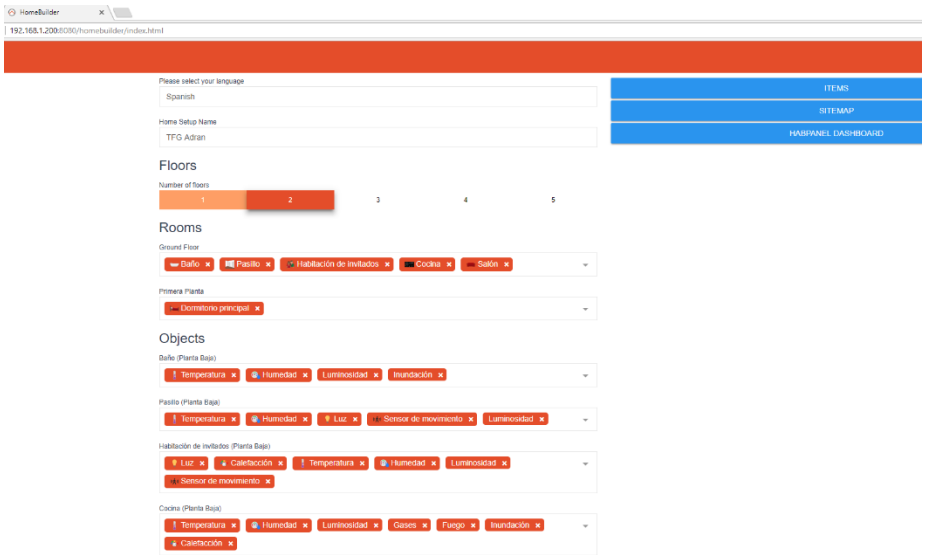
Configuración básica OpenHAB

Una vez iniciado el sistema, y desde el mismo explorador desde el que se había accedido por primera vez, se elige el paquete de instalación. En este caso será el paquete Standard, que es el recomendado por la plataforma. En la siguiente pantalla se puede observar que se instalan por defecto 4 interfaces diferentes:

- Home Builder:
- Basic UI:
- HABPanel:
- Paper UI:

Una vez instalado, se pueden añadir o desinstalar interfaces fácilmente desde PaperUI.

Lo primero que se realizará será crear una primera versión de las características del hogar desde Home Builder:



En los iconos azules de la derecha, se puede ver el código que se ha generado automáticamente, y el cual se tendrá que afinar manualmente. En el icono de HABPanel Dashboard se debe de copiar su contenido y pegarlo en el espacio de configuración de HABPanel. De la misma forma se hace click en los otros 2 iconos y se pega la información en un fichero de texto, puesto que servirá como base para la programación del sistema.

ANEXO III – Instalación bróker MQTT

I. Instalación de bróker MQTT

Se procederá ahora con la instalación y configuración básica del bróker MQTT Opensource Mosquitto. Para ello se usará el procedimiento de su web oficial [10] usando como apoyo información adicional [11].

Primero se accede a la Raspberry vía SSH y se lanzan los comandos de instalación, los cuales se han tenido que adaptar al estar trabajando con Stretch, una distribución de Raspbian que existe desde agosto de 2017, y sobre el cual no existía documentación actualizada.

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
sudo apt-key add mosquitto-repo.gpg.key
cd /etc/apt/sources.list.d/
sudo wget http://repo.mosquitto.org/debian/mosquitto-stretch.list
sudo apt-get update
cd ~
wget http://ftp.us.debian.org/debian/pool/main/libw/libwebsockets/libwebsockets3_1.2.2-1_armhf.deb
wget http://ftp.us.debian.org/debian/pool/main/o/openssl/libssl1.0.0_1.0.2l-1~bpo8+1_armhf.deb
sudo dpkg -i libwebsockets3_1.2.2-1_armhf.deb
sudo dpkg -i libssl1.0.0_1.0.2l-1~bpo8+1_armhf.deb
sudo apt-get install mosquitto
sudo apt-get install mosquitto-clients
```

Si no se ha recibido ningún error, como es el caso, se lanzan 2 comandos para comprobar el funcionamiento del sistema:

```
netstat -an --ip -p
```

```
pi@raspberrypi:/etc/mosquitto/ca_certificates $ netstat -an --ip -p
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:1883            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:5900            0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:37529         127.0.0.1:50562         ESTABLISHED 813/vncserverui
tcp        0    248 192.168.1.200:22        192.168.1.62:63927      ESTABLISHED -
tcp        0      0 127.0.0.1:50562         127.0.0.1:37529         ESTABLISHED -
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           -
udp        0      0 0.0.0.0:33130           0.0.0.0:*                           -
udp        0      0 0.0.0.0:68              0.0.0.0:*                           -
```

```
sudo systemctl status mosquitto.service
```

```
pi@raspberrypi:/etc/mosquitto/ca_certificates $ sudo systemctl status mosquitto.service
● mosquitto.service - LSB: mosquitto MQTT v3.1 message broker
   Loaded: loaded (/etc/init.d/mosquitto; generated; vendor preset: enabled)
   Active: active (running) since Tue 2018-03-13 18:42:22 UTC; 57min ago
     Docs: man:systemd-sysv-generator(8)
   CGroup: /system.slice/mosquitto.service
           └─3156 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

Mar 13 18:42:22 raspberrypi systemd[1]: Starting LSB: mosquitto MQTT v3.1 message broker...
Mar 13 18:42:22 raspberrypi mosquitto[3150]: Starting network daemon:: mosquitto.
Mar 13 18:42:22 raspberrypi systemd[1]: Started LSB: mosquitto MQTT v3.1 message broker.
```

Se observa que el Servicio está activo (running).

ANEXO IV – Instalación Samba

I. Instalación de servidor Samba

Como paso adicional, se instalará un servidor Samba [12] en la Raspberry para poder acceder a sus ficheros y modificarlos directamente mediante un editor enriquecido, en este caso mediante Notepad++ [13] con los plugin de Openhab, ya que la mayoría de las configuraciones se realizan vía fichero, y realizarlo en SSH es posible, pero mucho más tedioso, como se verá en los siguientes pasos en los que se modificará el fichero de configuración de Samba. De esta forma, se podrá usar un editor de texto enriquecido, que facilitará la detección de errores de sintaxis más fácilmente.

Para la instalación de Samba se siguen los siguientes pasos:

```
sudo apt-get install samba samba-common-bin
sudo nano /etc/samba/smb.conf
```

Y dentro del fichero smb.conf, se modifica lo siguiente:

Se descomenta la línea de wins support y se modifica a “wins support = yes”

Y se añaden las siguientes líneas al final del fichero para habilitar las carpetas especificadas en el path como accesibles y escribibles desde el servidor Samba:

```
[openHAB2-userdata]
  comment=openHAB2 userdata
  path=/var/lib/openhab2
  browseable=Yes
  writeable=Yes
  only guest=no
  public=no
  create mask=0777
  directory mask=0777

[openHAB2-conf]
  comment=openHAB2 site configuration
  path=/etc/openhab2
  browseable=Yes
  writeable=Yes
  only guest=no
  public=no
  create mask=0777
  directory mask=0777
```

Se cierra el editor (Ctrl+X, y pulsar “Y” para guardar los cambios), y se activa el usuario “OpenHAB” modificando su pw.

```
sudo smbpasswd -a openhab
```

Se comprueba que el usuario tiene acceso a las carpetas de configuración de OpenHAB:

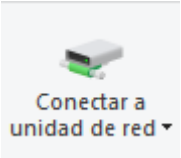
```
sudo chown -hR openhab:openhab /etc/openhab2
```

Y si no existe ningún error se reinicia el servidor con los cambios realizados:

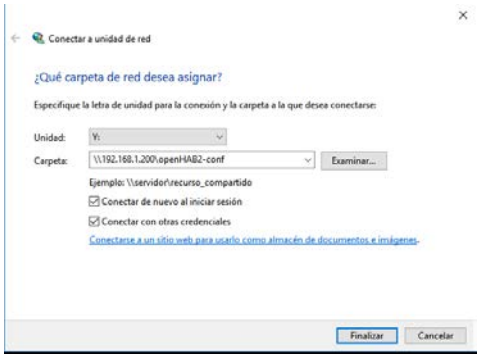
```
sudo systemctl restart smbd.service
```

El siguiente paso depende del sistema operativo con el que se trabaje. En este caso es Windows, pero se puede realizar de igual modo desde otro SO.

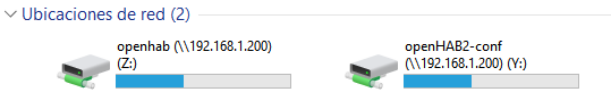
En Windows:

1. Se pincha en “Este Equipo”
2. Se hace click en “Conectar a unidad de red” 

3. Se asigna una letra de unidad, y en el apartado de Carpeta se introduce la dirección del servidor Samba, seguido de la carpeta a la que se quiere acceder. En este caso, openHAB2-conf, y openhab. Se marca la casilla “Conectar con otras credenciales”, puesto que, en caso contrario, el equipo intenta acceder con las del usuario de Windows en cuestión, y excepto que se hayan configurado las mismas (es posible hacerlo), la conexión fallará:



4. Se accede normalmente a ambas ubicaciones como si de carpetas locales en el propio equipo de tratasen.



Con esto, el sistema está listo para empezar a configurar los diferentes Bindings, las llamadas MQTT, y a configurar los Sitemaps, Things e Items del sistema.

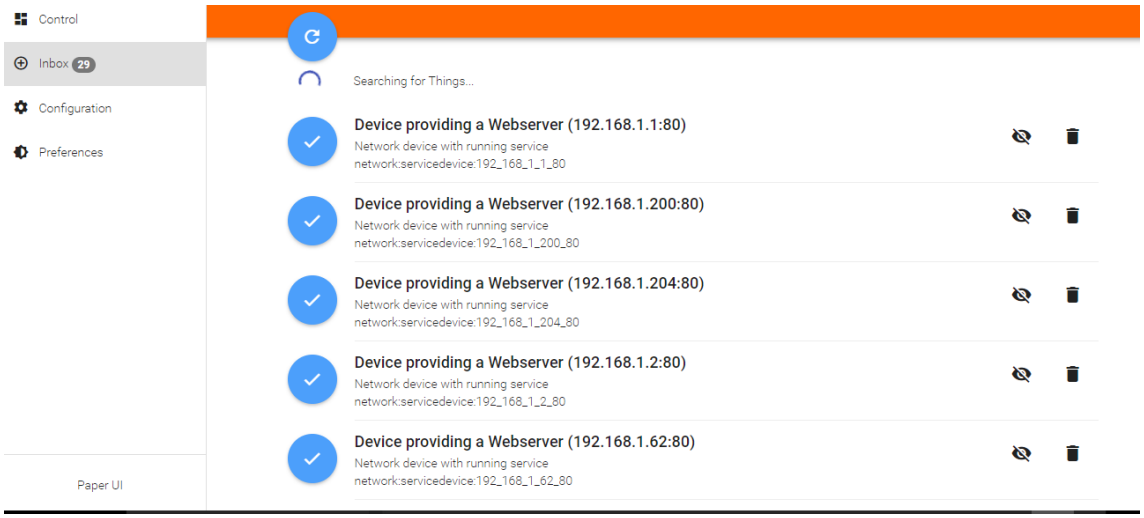
ANEXO V - Configuración de Openhab para conectividad

Una vez ya se dispone de Openhab, el Bróker MQTT, y el servidor Samba creados en la Raspberry, se puede comenzar a configurar Openhab [14] para que disponga de conectividad con el resto de la red local.

Lo primero será acceder a Paper UI-Configuración-Bindings, y añadir los siguientes Bindings:

- Network Binding: El cual realizará el autodescubrimiento de dispositivos de la red.

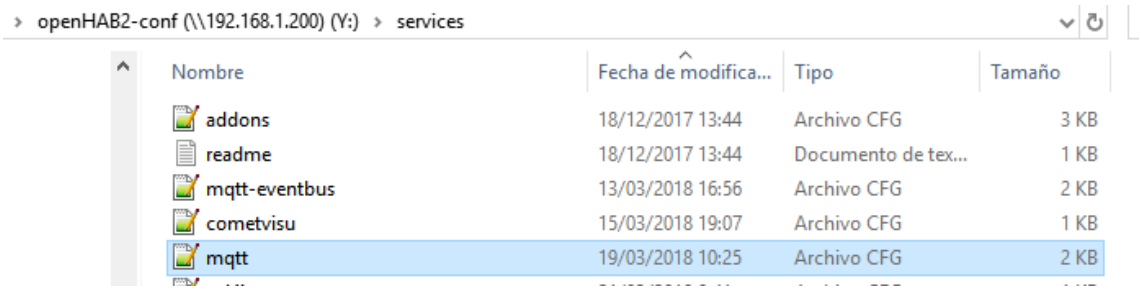
Si una vez instalado se accede a Inbox, se pincha en +, y luego sobre Network Binding, el cual descubrirá todos los dispositivos de la red local:



Se podrán añadir, y monitorizar. Normalmente se podrá ver si están Online, su latencia, y la fecha de última aparición en la red. Es una forma de monitorizar dispositivos, aunque no se pueda interactuar con ellos, actuando a modo de sensores (horas que está el ordenador encendido, cuando está un móvil conectado a la red local…)

- MQTT Binding: Con el cual se podrán realizar comunicaciones con el bróker MQTT, actuando Openhab como un cliente más del sistema.

Para configurar este Binding se accederá al fichero de configuración ubicado en la carpeta Services a través del servidor Samba:



Y se modificará el fichero des comentando las siguientes líneas, y cargando esta configuración:

```
broker.url=tcp://192.168.1.200:1883
broker.clientId=Openhab
broker.qos=1
broker.retain=true
broker.async=false
```

ANEXO VI – Servicios de persistencia (rrd4j, mapDB, InfluxDB)

Para poder realizar el mantenimiento de los datos, así como la recuperación de estos en caso de apagado (saber en qué estado se encontraba una luz), o graficar los diferentes valores de un Item (Temperatura a lo largo de un tiempo determinado), se tienen que establecer y configurar los servicios de persistencia, es decir, las BBDD en las que se mantendrá el histórico de la información.

En este sistema se implementarán servicios a través de 3 BBDD diferentes:

- Rrd4j: Para valores a corto plazo. Esta BBDD tiene un tamaño determinado desde el principio, por lo que almacenará valores minuto a minuto a corto plazo, pero con el paso del tiempo desecha la mayor parte de estos valores manteniendo incluso 1 valor por día para eventos de hace 1 año. Solo puede almacenar elementos numéricos

Para instalar esta BBDD en el sistema, puede realizarse desde Paper UI. Una vez instalada, se carga la siguiente configuración en /persistence/rrd4j.persist:

```
Strategies {
   everyMinute : "0 * * * * ?"
   everyHalf : "0 30 * * * ?"
   everyHour : "0 0 * * * ?"
   everyDay : "0 0 0 * * ?"
   default = everyChange, everyMinute
}

Items {
                * : strategy = everyChange, everyMinute
```

- mapDB: Esta BBDD solo tiene en cuenta el último valor que ha recibido para un elemento, por lo que es una BBDD apropiada para recuperar los valores de los Items en caso de apagado del sistema.

Para instalar esta BBDD se procede de la misma forma que en el caso anterior, pero esta vez realizando la siguiente configuración en /persistence/mapdb.persist:

```
Strategies {
       default = everyUpdate
}
Items {
// persist all items once a day and on every change and restore them from the db at startup
* : strategy = everyChange, restoreOnStartup
}
```

- InfluxDB: Por último, InfluxDB actuará como BBDD principal, puesto que permite exportar los datos a Grafana y realizar gráficas adaptadas a las necesidades de cada diseñador, no limitándose a las disponibles nativamente por Openhab:

Para instalar InfluxDB [15, 16, 17], también se debe de realizar la instalación sobre el sistema Openhab, y cargar la siguiente configuración en el fichero /persistence/Influxdb.persist:

```
Strategies {

    everyHour : "0 0 * * * ?"
    everyDay  : "0 0 0 * * ?"
    everyMinute : "0 * * * * ?"

     // if no strategy is specified for an item entry below, the default list will be used
    default = everyMinute
}

Items {
    // persist all items once a day and on every change
    * : strategy = everyMinute, everyChange, everyDay, restoreOnStartup
}
```

Además, se ha de crear el servidor de BBDD en la Raspberry, por lo que se ha de realizar el siguiente procedimiento por medio de comandos:

```
curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -
source /etc/os-release
test $VERSION_ID = "7" && echo "deb https://repos.influxdata.com/debian wheezy stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
test $VERSION_ID = "8" && echo "deb https://repos.influxdata.com/debian jessie stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
test $VERSION_ID = "9" && echo "deb https://repos.influxdata.com/debian stretch stable" | sudo tee /etc/apt/sources.list.d/influxdb.list

sudo apt-get update && sudo apt-get install influxdb
sudo systemctl daemon-reload
sudo systemctl enable influxdb.service
sudo systemctl start influxdb.service
```

Una vez la BBDD esté instalada y en servicio en el sistema, y habilitada para su autoarranque con el arranque del sistema, se crearán usuarios con diferentes perfiles y permisos en la nueva BBDD:

```
Influx
> CREATE DATABASE openhab_db
> CREATE USER admin WITH PASSWORD 'SuperSecretPassword123+' WITH ALL PRIVILEGES
> CREATE USER openhab WITH PASSWORD 'AnotherSuperbPassword456-'
> CREATE USER grafana WITH PASSWORD 'PleaseLetMeRead789?'
> GRANT ALL ON openhab_db TO openhab
> GRANT READ ON openhab_db TO grafana
> exit
```

A continuación, se realizarán los siguientes cambios sobre el fichero /etc/influxdb/influxdb.conf

```
[http]
enabled = true
bind-address = ":8086"      # change to a specific interface if needed
auth-enabled = true         # will enforce authentication
```

Guardando los cambios, y se restablece el servicio:

```
sudo systemctl restart influxdb.service
```

Se realiza la siguiente prueba de funcionamiento del sistema:

```
influx -username admin -password SuperSecretPassword123+ -host localhost
```

```
pi@raspberrypi:~ $ influx -username Openhab -password
Connected to http://192.168.1.200:8086 version 1.5.1
InfluxDB shell version: 1.5.1
```

Con esto, se tiene la BBDD de InfluxDB operativa y preparada para guardar valores de persistencia de Openhab.

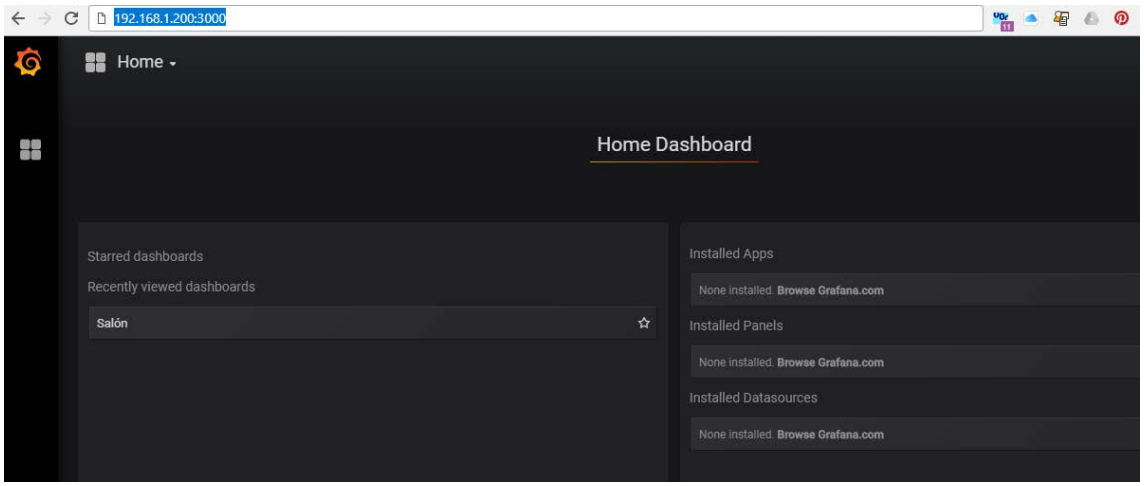
ANEXO VII – Grafana

Grafana es una aplicación open source que permite desarrollar cuadros de mandos sobre métricas basadas en series de tiempo, tanto para mostrar y analizar infraestructura (servidores, routers, sensores industriales, elementos domóticos, etc) como aplicaciones propias. En este caso, InfluxDB será el proveedor de datos de Grafana.

Para instalar Grafana [16, 17] se lanzarán los siguientes comandos en consola:

```
sudo apt-get -y update
sudo apt-get install -y apt-transport-https curl
sudo dpkg --add-architecture armhf
echo "deb [arch=armhf] https://dl.bintray.com/fg2it/deb jessie main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
curl https://bintray.com/user/downloadSubjectPublicKey?username=bintray | sudo apt-key add -
sudo apt-get -y update
sudo apt-get -y remove --purge grafana grafana-data
sudo rm -rf /var/log/grafana /etc/grafana
sudo apt-get -y update && sudo apt-get install -y grafana:armhf
sudo systemctl daemon-reload
sudo systemctl enable grafana-server
sudo systemctl start grafana-server
```

Se realiza la prueba lanzando http://192.168.1.200:3000 desde un explorador dentro de la red local para confirmar que el Servidor de Grafana está funcionando.



UOC

Universitat Oberta de Catalunya