UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Alcoy

DISEÑO Y CREACION DE UNA CASA DOMOTICA MEDIANTE ARDUINO Y RASPBERRY PI

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Beso Ballester, Valenti

Tutor/a: Masiá Vañó, Jaime

CURSO ACADÉMICO: 2021/2022

Resumen

Nuestro trabajo trata de encontrar una visión de la domótica y de los hogares inteligentes con un presupuesto bajo, los más completos posibles, utilizando hardware y software open-source. Para ello, iremos desde la programación de un sistema de seguridad completo, pasando por sensores de temperatura del hogar, hasta el control por voz del sistema al completo mediante el uso de Alexa como Inteligencia Artificial, usando un Arduino para el sistema físico y una Raspberry Pi para instalar el servidor que contendrá de openHAB.

Palabras clave: OpenHAB, Arduino, Raspberry Pi, Alexa

Resum

El nostre treball tracta de trobar una visió de la domòtica i de les llars intel·ligents amb un pressupost baix, els més complets possibles, utilitzant hardware i software open-source. Per a això, anirem des de la programació d'un sistema de seguretat complet, passant per sensors de temperatura de la llar, fins al control per veu del sistema al complet mitjançant l'ús d'Alexa com a Intel·ligència Artificial, usant Arduino per al sistema físic i una Raspberry Pi per a instal·lar el servidor que contindrà openHAB.

Paraules clau: OpenHAB, Arduino, Raspberry Pi, Alexa

Abstract

Our work tries to find a vision of home automation and smart homes with a low budget, as complete as possible, using open-source hardware and software. To do this, we will go from programming a complete security system, through home temperature sensors, to voice control of the entire system using Alexa as Artificial Intelligence, using an Arduino for the physical system and a Raspberry Pi to install the server that will contain openHAB.

Keywords: OpenHAB, Arduino, Raspberry Pi, Alexa

Agradecimientos

A mi madre y a mi padre, por comprar ese primer ordenador de segunda mano unas navidades, que me ayudo a hacer mis primeros movimientos en la informática.

Y a Isabella, que ha aguantado lo inaguantable mientras me apoyaba para la realización de este trabajo.

Tabla de contenido

Capítulo 1. Introducción

En esta introducción nombraremos los conceptos básicos tratados en esta memoria, los cuales serán explicados en profundidad durante las siguientes páginas. Explicaremos conceptos como el "Internet de las cosas" o IoT,

La idea llevada a cabo en esta tesis proviene de la necesidad y la curiosidad propia por el mundo de los microcontroladores y por el enorme potencial que tienen estas tecnologías para ayudar a las personas en el día a día. Según Statista[1], en 2020 casi un 50% de hogares españoles disponían de una Smart tv, lo que nos indica que la tecnología está asentándose en nuestras vidas sin ser conscientes de cuánto.

En esta tesis buscaremos soluciones económicas para convertir una casa en una casa inteligente, buscando el menor coste. Para ello, utilizaremos software de código abierto, también llamado open-source, y hardware con una buena relación calidad/precio.

1.1. Motivación

Nuestra motivación para la realización de este proyecto ha sido la búsqueda de la simplificación tecnológica en una casa y convertir una casa tradicional en una Smart house o casa inteligente.

Este proyecto no es nada nuevo, ya que existen ya multitud de soluciones en el mercado para llevar a cabo este fin, pero como hemos dicho anteriormente, usaremos código abierto el cual nos puede permitir tener mayor control de nuestra casa y podremos cambiar a nuestro antojo aquello que no nos guste, además de que supondrá un menor gasto para nosotros. Este tipo de tecnologías suponen un coste e inversión bastante elevada para un cliente, por lo que poca gente puede permitirse entrar en este campo. No obstante, muchísimas personas tienen elementos propios de una casa inteligente como puede ser una inteligencia artificial. Un claro ejemplo sería tener un HomePod de Apple, que integra a Siri en la casa, o un Google Home que integra a Ok, Google.

Tenemos gran cantidad de dispositivos inteligentes en ámbitos de nuestra vida muy distintos. Desde dispositivos para el hogar, pasando por las ciudades y la industria, edificaciones, etc. Podemos encontrar sensores que mejoran a trabajadores en el uso y manejo de maquinaria aumentando productividad, ciudades en las que podemos ver un uso eficiente de energía y servicios de movilidad viendo en tiempo real donde se haya dicho servicio o mejorando la sostenibilidad ahorrando en gasto energético.

Cada vez más empresas se suman al desarrollo y la investigación en este nuevo campo de la tecnología, ya que es una inversión a futuro, tanto para mejorar sus márgenes de beneficio al ser más eficientes como para los trabajadores optimizando su trabajo y mejorando sus puestos de trabajo.

Para los gigantes tecnológicos este mundo no pasa desapercibido. Microsoft ofrece los servicios "IoT de Azure" [2], Apple con sus productos [3] como los "Airtags" y su aplicación "Casa", IBM con su plataforma IBM Cloud, dispositivos y protocolos como MQTT [4].

La base fundamental del IoT son los datos recibidos por diferentes sensores y esos sensores. En España tenemos la Asociación Española de Domótica e Inmótica o CEDOM, la cual define la domótica [5] como "*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*."

La domótica se basa en el envío y recepción de datos desde sensores interconectados, estudiar esos datos y actuar acorde a ciertas normas que el usuario definirá según sus necesidades. Según la CEDOM, la domótica ayuda a la sostenibilidad al gestionar de manera inteligente la energía utilizada, fomenta la accesibilidad al ayudar en el manejo de elementos de una casa a personas discapacitadas, apoya la seguridad al instalarle cámaras de vigilancia accesibles desde otros dispositivos, garantiza las comunicaciones con la casa mediante el control remoto y un largo etcétera.

1.2. Objetivos

En este punto explicaremos los objetivos principales y las metas que nos hemos propuesto para este proyecto.

Como objetivo principal, nos hemos propuesto conseguir domóticas una casa de la manera más económica posible. Para ello utilizaremos microcontroladores como será un Arduino Mega y una Raspberry Pi con el sistema operativo Ubuntu Server. Para comunicar Arduino con Raspberry Pi usaremos una red de área local. Conectaremos por ethernet el Arduino Mega y por wifi la Raspberry Pi (también la llamaremos de ahora en adelante RPi). Para comunicarse entre ellas utilizarán el protocolo MQTT desarrollado por IBM.
Además, nos hemos propuesto implementar una inteligencia artificial open-source en la casa que interactúe con el Arduino mediante openHAB

1.3. Estructura de la memoria

A continuación, desglosaremos cómo va a estar estructurada la memoria del proyecto y explicaremos en qué consiste cada una de ellas.

1. **Introducción.**

   En este punto, hablaremos en líneas generales del contexto del proyecto, nombrando conceptos como puede ser IoT. Los explicaremos un poco y veremos la relación que guardan con nuestro proyecto.

2. **Estado del arte**

   Haremos un estudio de las posibles maneras y soluciones existentes para abordar nuestro proyecto y que podemos encontrar en el mercado. Las compararemos y expondremos ventajas e inconvenientes de cada uno de ellos, y explicaremos por que elegimos determinada plataforma para nuestro proyecto.

3. **Especificación de requisitos**

   Se presentarán los requisitos necesarios para llevar a cabo el proyecto y serán presentadas las características y funcionalidades del sistema. Veremos todas las partes implicadas en nuestro proyecto y serán explicadas.

4. Planificación y costes

   En este punto definiremos cada una de las fases de desarrollo y estimaremos el tiempo que deberíamos utilizar en cada una de las partes del proyecto. Detallaremos un cálculo aproximado del coste del proyecto

5. **Herramientas y tecnologías a utilizar**

   En este punto haremos un estudio de las tecnologías utilizadas y los protocolos que usaremos, explicando las herramientas que serán utilizadas para llevar a cabo este proyecto.

6. **Instalación y configuración**

   Aquí explicaremos como se instalará tanto el software como el hardware utilizado en el proyecto.

7. **Descripción grafica del entorno real y pruebas de rendimiento**.

   Definiremos el diseño con un plano de una casa genérica donde mostraremos donde se encuentran cada uno de los sensores distribuidos por la casa. Estudiaremos los resultados de las pruebas realizadas sobre la maqueta.

8. **Conclusiones**

   Explicaremos las conclusiones recogidas durante la realización de este proyecto y se intentara a posteriori incluir mejoras dentro del sistema.

9. **Referencias y bibliografía**

   En este apartado, encontraremos las referencias a documentación utilizada para la investigación y realización del proyecto.

10. **Anexos**

    En este último punto podremos ver el código utilizado para el Arduino y los archivos de configuración de OpenHAB.

1.4. Principales Referencias utilizadas

La envergadura de este proyecto es lo suficientemente grande como para tener que consultar documentación oficial. A continuación, podemos ver las referencias utilizadas para la realización de este proyecto:

- Documentación de la página web oficial de Arduino [5]. Si queremos informarnos de cómo utilizar correctamente este microcontrolador, el mejor sitio siempre será el oficial y estudiar esta documentación, nos servirá para utilizar como está pensada la placa.

- Documentación de la página web oficial de openHAB [6]. En su página web encontraremos una extensa documentación en varios idiomas y para sus diferentes

versiones del sistema, que nos ayudaran a configurar de manera óptima el proyecto acorde a las necesidades que tengamos en cada momento.

- Documentación de la página web oficial de MQTT [7]. En este sitio podemos encontrar tanto la explicación de como instalar este protocolo como la configuración de este.

- Documentación oficial de la web oficial de la biblioteca PubSubClient [15] donde encontramos la biblioteca PubSubClient, para entender cómo funciona dicha librería en Arduino para poder comunicar la RPi con el Arduino Mega mediante MQTT.

- Páginas de referencia de cada sensor. Hemos consultado cada una de las datasheets de los sensores para asegurar un correcto funcionamiento de estos.

- Documentación de la página web oficial de Mycroft IA [16]. Dado que esta inteligencia artificial es de Código libre, en su página tenemos una gran cantidad de información y documentación al respecto para una óptima instalación y configuración.

1.5. Relación con las asignaturas cursadas

Como ya hemos dicho, este trabajo se relaciona muy estrechamente con las asignaturas cursadas durante la carrera, las cuales expondremos de ahora en adelante y explicaremos en que parte están relacionadas con el proyecto.

Todas las asignaturas han sido importantes en este trabajo, aunque no se vean vinculadas de manera directa, ya que nos han enseñado una manera de trabajo o están relacionadas con materias importantes.

Siendo las más importantes o relacionadas todas las relacionadas con programación y con electrónica:

- Especificación de requisitos

Este punto se basará en un examen completo de las necesidades que tenemos para el caso expuesto, analizando en detalle las opciones que tendríamos y profundizando en las especificaciones requeridas. Las asignaturas que nos han ayudado a lograr dicho objetivo habrían sido Gestión de proyectos e Ingeniería del software, ya que nos enseñan como trabajar con una idea y desarrollarla hasta convertirla en una realidad.

- Diseño, planificación y elección de productos a utilizar

En este apartado encontraríamos asignaturas como Gestión de Proyectos junto a Modelos de Negocio y Áreas Funcionales que nos enseñaría como planificarnos, que recursos usaremos, como ahorraremos dinero, etc.

- Herramientas y tecnologías utilizadas.

Sistemas robotizados y Tecnología de computadores han sido esenciales en este apartado para la elección de las tecnologías disponibles, conocer y poder programar para ciertos tipos de placas de desarrollo diferentes y para finalizar, saber leer ciertas partes de una datasheet y conocer cómo actúan algunos sensores utilizados en este proyecto.

- Instalación y configuración del proyecto.

Nos encontramos ante el punto donde más asignaturas influyen, encontrándonos programación, Sistemas Robotizados, Tecnología de Computadores, Fundamentos de Sistemas Operativos, Concurrencia y Sistemas distribuidos y Redes de computadores. Encontramos Programación, Sistemas Robotizados y Tecnología de Computadores en la programación del Arduino, Concurrencia y Sistemas Distribuidos y Fundamentos de Sistemas Operativos en la programación y manejo de Ubuntu Server donde se encontrará OpenHAB y Mycroft AI, y Redes de Computadores para poder manejar las diferentes interfaces desde las que se controlará el sistema.

Capítulo 2. Estado del arte

En el siguiente punto estudiaremos el estado del arte, explicaremos y haremos una comparativa entre todas las posibilidades, exponiendo porque hemos elegido openHAB sobre las demás posibilidades.
Para las bases de este estudio presentaremos las posibilidades de código abierto que tenemos en el mercado.

2.1. Home Assistant

Esta plataforma de software libre es una manera de automatizar un hogar basado en python3. Esta opción es muy interesante al poder instalarlo en una RPi, ya que no es necesario que este en un servidor externo, aumentando la privacidad. Esta plataforma tiene una ventaja muy buena: Es multiprotocolo, muy personalizable y tiene una gran comunidad de usuarios. Aunque todo no es tan bonito, ya que las actualizaciones de las que dispone son inestables

2.2. openHAB

Nos encontramos con un software de código abierto cuyo propósito es integrar sistemas y tecnologías de automatización de viviendas. Esta es una muy buena opción, ya que, en este sentido, el sentido de esta plataforma es aunar cualquier tipo de dispositivo, independientemente de su procedencia, ya que es una plataforma multiprotocolo. Además, existe su aplicación móvil desde la que se nos permite ver y controlar el estado de los sensores y alterarlos a nuestro gusto.

2.3. Jeedom

En esta ocasión tenemos un software de código libre en el cual, teniendo nociones básicas de Linux, como en las anteriores opciones, podríamos configurar todo a nuestro gusto sobre una plataforma como una Raspberry pi.
En el caso de no disponer de un hardware, el desarrollador vende en su página web sus controladores a la venta para que sea más fácil de configurar para los usuarios poco acostumbrados al mundo de la domótica y la programación.
Lo que nos ofrece Jeedom resulta una opción muy atractiva, ya que nos ofrecen una imagen del sistema a coste 0, y dispositivos con la imagen ya instalada. El precio de la versión para uso doméstico está alrededor de unos 220 euros mientras que la versión profesional ronda unos 900 euros.
Esta opción nos ofrece un extra de seguridad, soporte técnico, ahorro de energía, avisos por SMS, comunicación por voz además de cómo hemos dicho antes controladores hechos por ellos mismos.
Como inconveniente, nos encontramos que gran cantidad de plugins para poder hacer funcionar ciertos aspectos de la parte domótica son de pago, y su precio ronda entre los 2 y los 6 euros.

2.4. Alternativas

En el mercado, podemos encontrar gran cantidad de alternativas, aunque sólo nombraremos brevemente algunas, siendo también muy buenas opciones a la hora de domotizar una casa.

- Apple Home.

La aplicación de Apple Home nos permite manejar gran cantidad de, como ellos mismos llaman, accesorios. Desde aires acondicionados, pasando por cerraduras, luces y enchufes hasta puertas de garaje, humidificadores, etc.

Un inconveniente de este sistema es la necesidad de un ecosistema propio de Apple para sacarle el máximo partido, ya que muchos de los accesorios, no son compatibles con otros sistemas operativos móviles.

Nos encontramos también con la barrera del precio, ya que son electrodomésticos con un precio bastante elevado frente a los sensores y electrodomésticos que usaremos nosotros.

- Xiaomi Smart Home

Nos encontramos en este caso con un kit bastante asequible de la empresa china Xiaomi. Aunque es una alternativa bastante económica, tiene limitaciones tanto por legislación española como por barrera lingüística. La aplicación para manejar los diferentes sensores esta principalmente en chino, y las pocas traducciones que tiene la aplicación son bastante básicas. Por otro lado, no está comercializado en España por no dar una garantía de 2 años. No permite integración con servicios como el nombrado anteriormente de Apple o Google Home, con lo que no podremos usar Alexa o Siri.

La elección de OpenHAB fue la que más se acercaba a lo que más se acercaba a los resultados esperados. Las dos opciones con más peso de todas las barajadas fueron OpenHAB y Home Assistant, descartando la última por la gran cantidad de actualizaciones que sufre el sistema, siendo más inestable que OpenHAB.

Capítulo 3. Especificación de requisitos

Este punto tiene como objetivo la definición del proyecto y las funciones a suplir por este, adaptándose lo máximo posible a las necesidades de los usuarios. A continuación, definiremos las funcionalidades del sistema.

3.1. Alcance

Los costes de estos proyectos suelen ser elevados y la gente neófita en la materia tiene miedo de tener que hacer grandes reformas para poder conseguir una casa completamente domotizada. Este proyecto propone una manera económica y sin grandes reformas, de tener una casa totalmente domotizada.

El sistema ira cableado por las cañerías ya instaladas de la casa, solucionando gran parte del problema de las reformas para la instalación de los sensores.

3.2. Perspectiva del sistema domótico

Nuestro sistema domótico se basará en la idea de ser accesible en la mayor cantidad de dispositivos posibles. Para ello, buscaremos que se pueda acceder desde el navegador de un ordenador o smartphone, además que buscaremos que tenga una aplicación nativa para que todo pueda ser controlado y monitoreado.

En cuanto a la conexión de las diferentes partes implicadas, tendremos la mayor parte del proyecto cableado, aunque también podríamos tener el servidor central conectado de manera inalámbrica, y usaremos el protocolo MQTT para enviar la información de los sensores hasta la centralita.

3.3. Tipos de dispositivos.

Los dispositivos que conforman este sistema domótico serán los descritos en brevedad a continuación:

- **Servidor Central**.

  Este será la pieza central que una todas las partes: Arduino, sensores, OpenHAB y Mycroft. En este servidor central tendremos un Ubuntu Server adaptado ya que el hardware utilizado para ello será una Raspberry Pi, en el que tendremos una instancia de openHAB y Mycroft en ejecución.

- **Centralita domótica**.

  Se compondrá de un Arduino mega que hará de intermediario entre los sensores y el servidor central. Controlará la parte domótica de manera independiente y mediante MQTT, openHAB también podrá controlar y darle órdenes para controlar y recoger información de los sensores.

- **Sensores y activadores**.

  Tendremos varios tipos de sensores como sensores RFID, sensores de gas, de ultrasonidos o de temperatura y de humedad.
  En cuanto a actuadores tendremos interruptores táctiles, matrices numéricas, LCDs y relés.

  Con todos estos, seremos capaces de monitorear y controlar el estado. Del hogar en tiempo real mediante el protocolo MQTT

3.3.1. Funcionalidad del sistema domótico

Nuestro sistema, aunque sea ampliable, ofrecerá unas mínimas comodidades al cliente como ahorro de energía al poder apagar ciertos electrodomésticos de la red eléctrica, seguridad frente a intrusiones en el hogar y comodidad ante imprevistos como apagar luces encendidas estando fuera de casa.

Encontraremos las siguientes funcionalidades en nuestro sistema domótico:

Función	Sensor/Actuador	Descripción
Temperatura	DHT11	Medirá la temperatura de la estancia donde se encuentre el sensor
Humedad	DHT11	Medirá la humedad relativa de la estancia
Alarma	Teclado Matricial 4x4 HC-SR501	Activara una alarma si no es desactivada correctamente
Seguridad de la puerta principal	RC522	Controlará la apertura de la puerta principal del domicilio.
Control de Persianas	Módulo relé 4 canales	Controlará la subida o bajada de las persianas
Calentador / Calefacción	Módulo relé 1 canal	Encenderá o apagará la calefacción o calentador (dependiendo para que lo programemos)
Luces	TTP223B	Controlaremos las luces, incluido desde el móvil
Estudio de Gases	Sensor MQ-2 Sensor MQ-7	Llevaremos un estudio de si hay fuga de gases en la cocina del hogar, asegurando la seguridad del residente
Control por voz	Alexa Dot 3	Controlaremos el resto de las funciones ya descritas por comandos de voz
Sensor de Aparcamiento	HC-SR04	Controlaremos si la plaza de aparcamiento esta libre u ocupada por algún vehículo.

Además de que todas estas funciones, y todas aquellas que queramos añadir, será posible que sean controladas con el móvil por su aplicación nativa o mediante un navegador web.

No añadiremos en esta lista algunas de las funcionalidades como un hilo musical, ya que estaría incluido en el control por voz

3.3.2. Restricciones

Para este proyecto, tendremos ciertas restricciones a considerar a la hora de proceder. Tendremos que suplir las necesidades del proyecto teniéndolas en cuenta.

Diseñaremos el sistema teniendo en mente su escalabilidad, haciendo que añadir más dispositivos a él sea sencillo. Además, todas las funciones deberán estar disponibles para todos los usuarios.

Los dispositivos conectados deben tener una comunicación rápida con el servidor y segura. Para ello, deberemos tener una conexión bien cableada entre los diferentes elementos del sistema.

Una de las restricciones más fuertes para que este sistema este siempre conectado seria la electricidad ya que, si sufriésemos un apagón, estaría fuera de nuestro alcance solucionarlo.

Capítulo 4. Planificación y costes

A continuación, definiremos los parámetros y las fases que llevaremos a cabo para la construcción de nuestro proyecto. Detallaremos, además los tiempos que hemos destinado a cada fase, los costes y los materiales utilizados para ello.

4.1. Descripción de las fases

En este punto describiremos cada una de las fases señalando los puntos más importantes de cada fase y el tiempo invertido en cada una.

- Fase 1. Alcance y definición del proyecto.

En este punto planteamos el alcance que tendrá el proyecto y estudiamos las posibilidades que tenemos para realizar el proyecto.
La decisión final ha sido utilizar openHAB utilizando un Arduino, una Raspberry Pi y los protocolos MQTT.

- Fase 2. Definición de los requisitos.

Acabados los preparativos y los estudios del alcance del proyecto, continuaríamos con las funciones de las que contará el sistema. Probaremos las herramientas y nos familiarizáremos con ellas.

- Fase 3. Estudio de sistemas alternativos.

Realizaremos un estudio de las alternativas disponibles en el mercado y las compararemos para encontrar cual se ajustará más a nuestras necesidades.

- Fase 4. Pruebas con las herramientas utilizadas.

En esta fase realizaremos un estudio de las funciones, los protocolos y herramientas que utilizaremos en el proyecto. Utilizaremos para ello las documentaciones oficiales, datasheets y ejemplos en las páginas oficiales de cada sistema.

- Fase 5. Diseño Hardware y Software.

Definiremos cada uno de los elementos que vamos a utilizar en el proyecto, tanto hardware como software.

- Fase 6. Configuración y Instalación.

Montaremos el sistema, instalaremos y configuraremos el servidor que utilizaremos para el sistema domótico. Esta fase será más larga ya que en ella debemos crear el código, instalar y configurar el servidor, y optimizar el código para su correcto funcionamiento.

- Fase 7. Descripción del entorno, automatizaciones y pruebas.

Diseñaremos un plano de la vivienda, colocando cada uno de los sensores en cada habitación. También mostraremos las automatizaciones del sistema domótico y realizaremos pruebas del funcionamiento.

- Fase 8. Realización de la memoria.

Redactaremos la memoria final, recogiendo cada una de las notas tomadas durante la realización del proyecto, revisando, mejorándolas y condensándolas para que aporten sentido al proyecto, y explicando cada una de ellas, de modo que tomen sentido en la memoria.

- Fase 9. Bibliografía y revisión de referencias.

Para esta última fase, revisaremos las faltas ortográficas y las referencias bibliográficas, comprobando que son correctamente referenciadas y la información es la correcta.

Después de explicar todas las fases que compondrán el proyecto, podemos realizar un estudio del tiempo que hemos usado para cada fase del proyecto, el estudio de los lenguajes de programación utilizados, la implementación de estos lenguajes y el sistema estudiado en el proyecto y en la documentación de este.

En la siguiente tabla, veremos el tiempo usado en cada una de estas fases.

Horas invertidas en las diferentes etapas	
Fases	**Tiempo**
Fase 1. Alcance y definición del proyecto	15 horas
Fase 2. Definición de los requisitos	13 horas
Fase 3. Estudio de sistemas alternativos	27 horas
Fase 4. Pruebas con las herramientas utilizadas	68 horas
Fase 5. Diseño Hardware y Software	18 horas
Fase 6. Configuración y Instalación	182 horas
Fase 7. Descripción del entorno, automatizaciones y pruebas	14 horas
Fase 8. Realización de la memoria	94 horas
Fase 9. Bibliografía y revisión de referencias	14 horas
Total de Horas	445 horas

4.2. Recursos

Llegados a este punto, debemos nombrar tanto los recursos de hardware como de software, usados en este proyecto.

4.2.1. Hardware

- Raspberry Pi 4 Model B 8GB
- 2 Arduino Mega
- Ethernet Shield
- MacBook Pro (15-inch, 2018), Intel Core i7 de seis núcleos a 2,2 GHz, 16 GB RAM 2400 MHz DDR4
- Ordenador de Sobremesa, con procesador Intel Core i7 de seis núcleos a 2,2 GHz, Memoria RAM Kingston HyperX Fury DDR4 2133 PC4-17000 8GB CL14 x2 (16GB)
- Alexa Dot 3
- Diversos LEDS de diferentes colores

4.2.2. Software

- Windows 10, utilizado en el ordenador de sobremesa para programar la placa de Arduino y configurando e instalando el sistema utilizado en la Raspberry Pi
- macOS Monterey, utilizado en el ordenador de sobremesa para programar la placa de Arduino y configurando e instalando el sistema utilizado en la Raspberry Pi
- Ubuntu Server LTS, utilizado para instalar el server utilizado en la Raspberry Pi 4
- OpenHAB 2.0
- IDE Arduino para programar la placa Arduino MEGA
- El programa PuTTY para conexiones SSH desde sistemas con Windows como SO principal.
- iOS 15.0 para la instalación de la aplicación openHAB

4.3. Proyección de costes

A continuación, vamos a realizar un estudio de los costes del Hardware y el software empleados en el desarrollo de nuestro proyecto. Debido a que estamos hablando de software y hardware libre, los costes son más bajos respecto a las opciones planteadas. Aquí tenemos una lista detallada de los costes del producto final que entregaremos al cliente.

Productos	Unidades	Precio
Raspberry Pi Modelo 4 B 8GB	1	88€
Shield Ethernet	2	14€
Arduino Mega	2	42€
Sensor DHT11	2	3€
Modulo relé 4 canales	1	4€
Modulo relé 1 canal	2	2€
Interruptor Táctil TTP223B	6	1€
Sensor PIR	1	3€
Modulo Buzzer	1	1€
Caja de LED 5 colores	1	7€
Sensor de gas MQ-2	1	3€
Sensor de gas MQ-7	1	4€
Sensor HC-SR04	1	3€
Teclado Matricial 4x4	1	2€
LCD 1602 + Interfaz I2C	2	5€
Modulo RFID RC522	1	4€
Amazon Echo Dot 3	1	28€
TOTAL		285 €

Cabe destacar que después de la realización del trabajo, hemos advertido que Mycroft debido a ciertas limitaciones de hardware que no podemos conseguir se ha hecho difícil continuar con esa parte de nuestro proyecto y hemos decidido cambiarlo por Amazon Alexa ya que su implementación en el proyecto es más sencilla.

Capítulo 5. Herramientas y tecnologías utilizadas

Vamos a explicar en este capítulo las tecnologías y herramientas que vamos a utilizar en nuestro proyecto. También explicaremos los protocolos utilizados para el correcto entendimiento del funcionamiento del proyecto.

1. Características de la Raspberry Pi 4B
2. Características del Arduino Mega
3. Características y definición de OpenHAB
4. Definición del protocolo MQTT
5. Características de Amazon Alexa

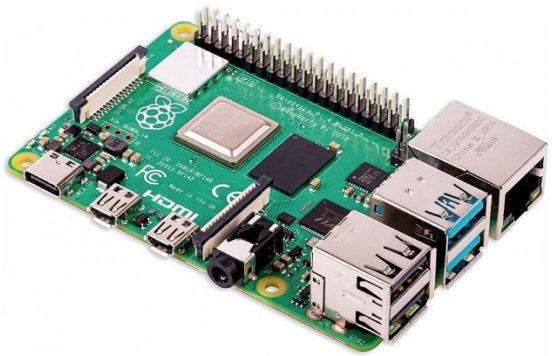
5.1. Raspberry Pi

Raspberry Pi es un proyecto open-source creado para hacer más fácil la entrada en el mundo de la informática y la programación. Se trata de un miniordenador de placa única del tamaño de una tarjeta de crédito. Este proyecto viene de la Raspberry Pi Foundation, una fundación de caridad de UK que aspira a que más gente se interese por la programación.
Estas placas han ido mejorándose modelo tras modelo hasta que de tener una CPU single-core a 700Mhz y 256 Mb de RAM a un modelo con CPU a 1,5 GHz de 64 bits y 8 Gb de RAM.

El sistema operativo que nos proporcionan en su página web como recomendado es Raspbian, una imagen de Debian modificada y optimizada para que funcione correctamente en la Raspberry pi. No es el único SO que soporta la placa, ya que cualquier sistema Linux modificado para un ordenador de bajos recursos funcionaria.

5.1.1. Raspberry Pi Model 4 B

Hablamos de una placa con un procesador ARM Broadcom BCM2711 de 64 bits a 1,5 GHz de velocidad, 8 GB de RAM a 2400MHz, conectividad Bluetooth 5.0, WIFI 802.11ac a 2,4 y 5 GHz, conexión Gigabit Ethernet, 2 puertos UB 2.0 y 2 puertos más 3.0, cabezal GPIO de 40 pines, 2 puertos microHDMI, un puerto USB-C de carga, puerto de cámara MIPI CSI de 2 carriles, puerto de visualización MIPI DSI de 2 carriles, gráficos OpenGL ES 3.0 y una ranura micro-SD donde tendremos el SO y el almacenamiento de datos.



5.2. Arduino

Arduino, según está definida en su página web, es una empresa que diseña, fabrica y admite dispositivos electrónicos y software, lo que permite a las personas de todo el mundo acceder fácilmente a tecnologías avanzadas que interactúan con el mundo físico. [6]

Una de sus máximas como empresa es que todos sus productos son open-source, con lo que cualquier empresa o particular puede hacer cambios en su diseño con la documentación correspondiente, crear sus propias placas, etc. Esto implica que, aunque sean diferentes entre ellas, parten de la misma base e idea original.

Las placas Arduino están basadas en un microprocesador ATMEL, que interactúa con los diferentes dispositivos que conectemos a sus puertos de Entrada/Salida (de ahora en adelante E/S), aportando datos que podremos utilizar en nuestros proyectos, o dándonos un sinfín de posibilidades para aprender robótica/domótica.

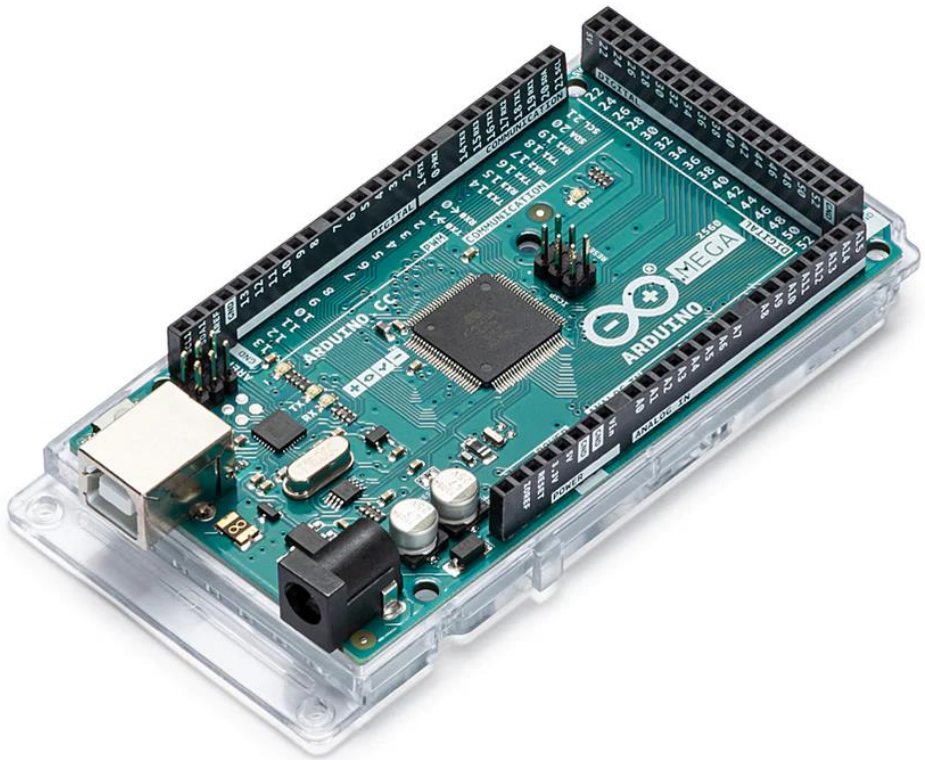
Realmente Arduino es un proyecto más que un modelo de placa. Es decir, partiendo de la misma base, cada modelo es diferente entre sí. Ya sea porque un modelo tiene más pines de E/S, ya sea porque tiene una conexión de ethernet integrada a la placa, cada modelo tiene unas particularidades que se adaptaran mejor a cada uno de los posibles proyectos. Una de las mejores cosas que tiene este proyecto es que cualquier placa sirve para casi cualquier proyecto gracias a los "Shields" o placas de expansión. Estos Shields le aportan a Arduino funcionalidades extra para que se pueda trabajar con aspectos para los que no estaba diseñado desde el principio pero que son capaces de realizar. Por ejemplo, el Arduino UNO es la placa más básica de todas y por la que un neófito en la materia puede empezar a aprender. Esta no tiene puerto ethernet, pero si le añadimos un Ethernet Shield podemos convertir nuestro Arduino UNO en un servidor web funcional.

Además, contamos con el respaldo de una comunidad muy grande de "makers" que, en sus propios foros, pueden ayudarnos a encontrar soluciones o consejos para poder llevar a cabo nuestro proyecto.

5.2.1. Arduino MEGA

En su página web, además de encontrar esquemas de diseño electrónico y sus datasheets, también encontraremos una tabla con sus características.

En esta placa tendremos un ATmega2560 a 16MHz de velocidad de reloj, funcionando a un voltaje de 5V, aunque el voltaje al que debe ser conectado oscila entre 7 y 12 V, con 54 pines de E/S digitales, de los cuales 15 serán de PWM, y 16 pines analógicos. también tendremos una diferencia en las capacidades de las memorias. El Arduino MEGA constara de 256 KB de memoria flash, usados 8 KB por el bootloader, 8 KB de SRAM y 4 KB por la EEPROM.



5.3. Proyecto openHAB

Nos encontramos ante un proyecto open source desarrollado en Java y que está basado en Eclipse Smart home. Uno de sus principales atractivos es su diseño, ya que es independiente del proveedor funcionando así cualquier protocolo de comunicación y cualquier hardware.

Uno de los principales inconvenientes del internet de las cosas es la interoperabilidad. Muchos de los productos en el mercado no se pueden combinar entre sí haciendo difícil crear un ecosistema entre sensores de varios fabricantes. cada fabricante utiliza un diseño y un protocolo diferente a la hora de comunicarse con un servidor domótico. Por este motivo surge OpenHAB.

openHAB, gracias a las opciones de configuración que tiene, nos permite utilizar cualquier dispositivo sin importar fabricante ni protocolo utilizado para sus conexiones, dándonos así la libertad de poder hacer uso de cualquier solución para nuestro sistema domótico.

Este proyecto no surge para desbancar a la competencia sino para mejorar las opciones ya existentes mejorando la seguridad el funcionamiento y la integración entre los diferentes tipos de sensores y actuadores que existen en el mercado.

Uno de sus puntos fuertes es la cantidad de bindings configurables ya que facilita mucho el poder configurar un dispositivo como por ejemplo una tira de leds de una marca en específico, un Amazon Alexa como asistente domótico, etc.

Cabe destacar que para utilizar estos comandos el usuario medio tiene que leerse la documentación completa para poder entender el funcionamiento y la configuración necesaria que se adapte a sus necesidades, invirtiendo así una cantidad de tiempo considerable.

Llegados a este punto, es necesario que queden definidos ciertos conceptos como “things”, “ítems” o “channels”. Estos conceptos son explicados en más detalle en la documentación oficial de openHAB, pero describe las “things” como todo aquello que se pueda añadir al sistema de manera física y desarrolle alguna funcionalidad. Con esto englobaríamos también servicios web, conexiones, etc. Serian aquellas cosas que nos proporcionen información del mundo exterior.
Cada thing disfrutara de uno o varios “channels”. Un channel sería la manera que tiene el sistema de relacionarse con cada una de las fuentes de información. Y por último, dentro de cada channel tendríamos los ítems, que serían cada una de las funcionalidades que tendría el sistema, siendo capaces de dar información y cambiar de estado ya sea mediante reglas o la interfaz de usuario. Para llegar a relacionarse todas estas entidades, dependeremos de los enlaces o “bindings”

5.4. Protocolo MQTT

El nombre de este protocolo son las siglas Message Queing Telemetry Transport. Hablamos de un protocolo de comunicación M2M de tipo message queue, basado en la pila tcp/ip como base para la comunicación

El funcionamiento de este protocolo es un servicio de mensajería pulse con patrón publicador /suscriptor, es decir, un cliente publica un mensaje en un canal y un suscriptor lee los mensajes de ese canal al que debe estar previamente suscrito.

Para este tipo de conexión, necesitamos un servidor central llamado bróker, que se encargara de organizar jerárquicamente los topics y hacerle llegar a cada uno de los suscriptores de cada canal los mensajes publicados en el canal correspondiente.

Un ejemplo práctico sería el siguiente:
Tenemos un sensor que publica la temperatura. Este sensor le enviara al bróker su temperatura en un canal llamado “temp”. Para poder leer esa temperatura, el suscriptor deberá suscribirse a el canal “temp” y el bróker le enviará la información filtrándola de otra información que este en canales diferentes. Es decir, solo le enviara los paquetes de información que estén en el canal “temp”.

Uno de los principales atractivos es que un mismo sensor puede publicar diferente información a la vez y enviarla a su respectivo canal, haciendo posible que la información sea visualizada de manera concreta por diferentes suscriptores.

Trasladado al ejemplo anterior, podríamos tener un sensor que publique la temperatura y la humedad, en un canal diferente cada información, y que un suscriptor lea solo la temperatura de un canal y otro suscriptor lea la humedad de otro canal diferente.

A continuación, describiremos los aspectos más relevantes del funcionamiento de este protocolo.

Los clientes iniciarían una conexión TCP/IP, como ya hemos dicho anteriormente, mediante el puerto 1883 por defecto o el 8883 di funciona sobre TLS, manteniendo la conexión abierta hasta que el cliente la finalice.

Para esto, el cliente enviara al bróker un mensaje CONNECT con la información necesaria, y este le responderá con un mensaje CONNACK informándole de la resolución de la conexión, ya haya sido aceptada o no.

Si ha sido aceptada, el publicador enviara mensajes PUBLISH indicando el canal (de ahora en adelante topic) y la información (de ahora en adelante payload).

Por otro lado, el suscriptor para poder leer esta información deberá enviar mensajes SUBSCRIBE en los que el bróker le contestará con un SUBACK, o desuscribirse con un mensaje UNSUSCRIBE que le devolverá un UNSUBACK el bróker.
Para acabar y como dijimos anteriormente, el cliente es el que cerrara la conexión enviando un mensaje de DISCONNECT.

Otro aspecto relevante de este protocolo es el mecanismo de calidad de servicio (QoS). Este mecanismo es el que se encarga de gestionar y verificar frente a posibles fallos que el mensaje es recibido. Tenemos para ello 3 niveles de QoS:

QoS 0: Conocido como “máximo una vez”. El mensaje solo se envía una vez, pudiendo no entregarse el mensaje si hay algún fallo.

QoS1: “Al menos una vez”. Se envía el mensaje hasta que se garantiza la entrega, pudiendo el suscriptor recibir duplicados algunos mensajes.

QoS: “Exactamente uno”. Se garantiza la entrega al suscriptor del mensaje una única vez.

Este mecanismo nos proporciona unas medidas de fiabilidad, dependiendo de las necesidades de nuestro sistema o proyecto.

También habría que destacar la seguridad de este protocolo, ya que dispone de distintas medidas de seguridad que solo nombraremos para no extendernos demasiado: Dispondremos de transporte SSL/TLS, autentificación por usuario y contraseña o mediante certificado.

Todas estas características, además de las ventajas como pueden ser su sencillez, su ligereza, su bajo consumo de energía y la seguridad hacen que en equipos de pocos recursos funcione a la perfección, además de tener años de funcionamiento a sus espaldas por lo que tiene un buen respaldo por parte de la comunidad.[7]

5.5. Amazon Alexa

Según Amazon [8], “Alexa es el servicio de voz ubicado en la nube de Amazon disponible en los dispositivos de Amazon y dispositivos tercios con Alexa integrada.”

Entendemos por Alexa, los productos estrella de Amazon: los altavoces Dot. Aunque realmente, como hemos explicado anteriormente, Alexa es la inteligencia artificial diseñada por Amazon, y que si está integrada en estos dispositivos. Cabe destacar que Alexa cada vez esta más presente en nuestra vida: ya sea cuando instalas los controladores de un ratón Razer, ya sea porque lo instalamos en nuestro teléfono, al final podemos controlar casi cualquier dispositivo gracias a ella.

Es por ello que, gracias a esta inteligencia artificial, la desarrolladora que la avala y que openHAB tiene una skill que permite relacionar estos dos ecosistemas, hemos decidido utilizarla finalmente para nuestro proyecto.

Tenemos unas ventajas respecto a la inteligencia artificial anteriormente nombrada para este proyecto, que no podemos pasar por alto como puede ser la facilidad para tenerla en nuestros dispositivos de uso diario, el idioma de uso, los pocos recursos que necesitamos para tenerla cerca de nosotros, e incluso el nivel de detalle que recoge de nuestras palabras o el margen de error de entendimiento que tiene que es bastante bajo. Mas adelante detallaremos más extensamente por que se ha decidido usar esta inteligencia artificial y hemos decidido prescindir de la otra.

Capítulo 6. Instalación y configuración

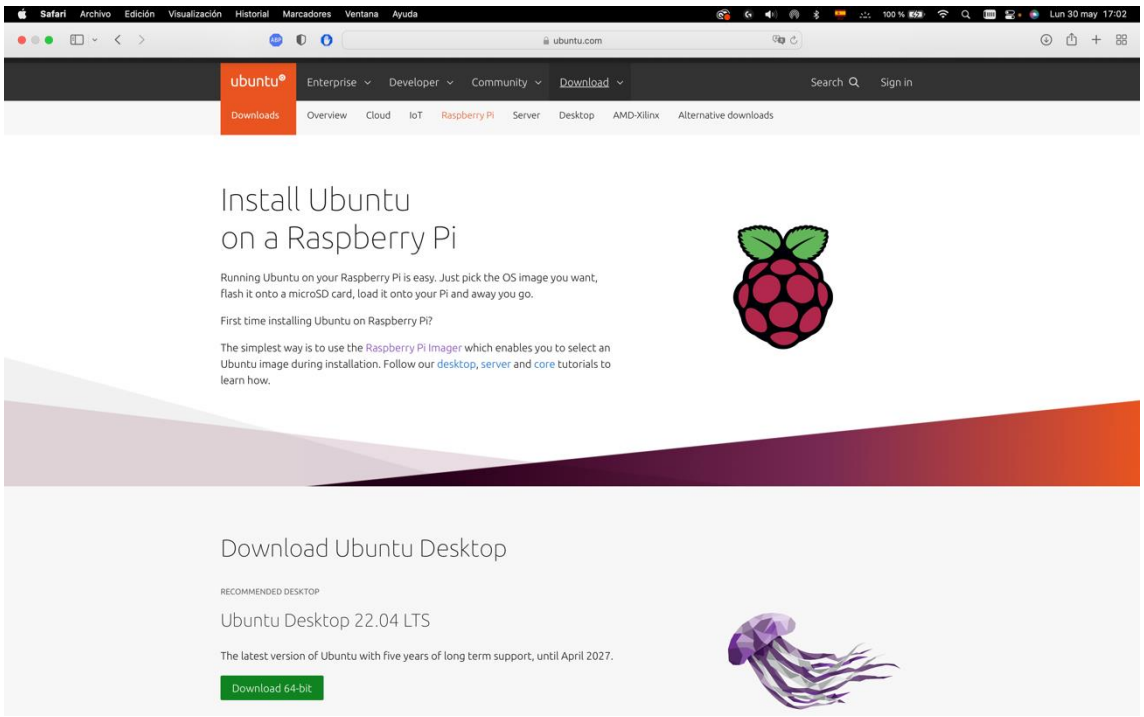
En este capítulo, instalaremos paso a paso y configuraremos el ecosistema completo: Tanto sensores, como Arduino, como la instalación y configuración de openHAB en nuestra Raspberry Pi.

Para ello, deberemos descargar los recursos necesarios de las respectivas páginas web. Primero instalaremos un Ubuntu Server modificado en nuestra Raspberry pi, realizaremos la instalación de openHAB, y Mycroft.

Configuraremos cada uno de los aspectos necesarios como el bróker MQTT necesario, instalación y configuración en openHAB de las skills necesarias para el funcionamiento de Mycroft y posteriormente Alexa, y enseñaremos como se configuran en cada uno de sus entornos.

6.1. Configuración Raspberry Pi

En nuestro proyecto, instalaremos una versión modificada de Ubuntu Server provisto en la propia página de Ubuntu. En nuestro caso, usaremos Ubuntu server 22.04.3 TLS.



Una vez descargado usaremos balena etcher para grabar la imagen en una memoria micro-SD y esperaremos a su completa grabación. Al finalizar, la introduciremos en la Raspberry pi y esperaremos a que se configure ella sola. El proceso puede tardar unos minutos.

Al tener la Raspberry conectada por un cable ethernet, para configurarla por SSH deberemos acceder al router y averiguar que IP se le ha asignado a nuestra raspberry. Como estamos

trabajando en dos redes diferentes, tendremos dos IP diferentes durante el transcurso del proyecto. Las IP serán 192.168.0.24 y 192.168.1.146.
Al acabar de instalarse y acceder mediante SSH, accederemos con la siguiente instrucción desde terminal:

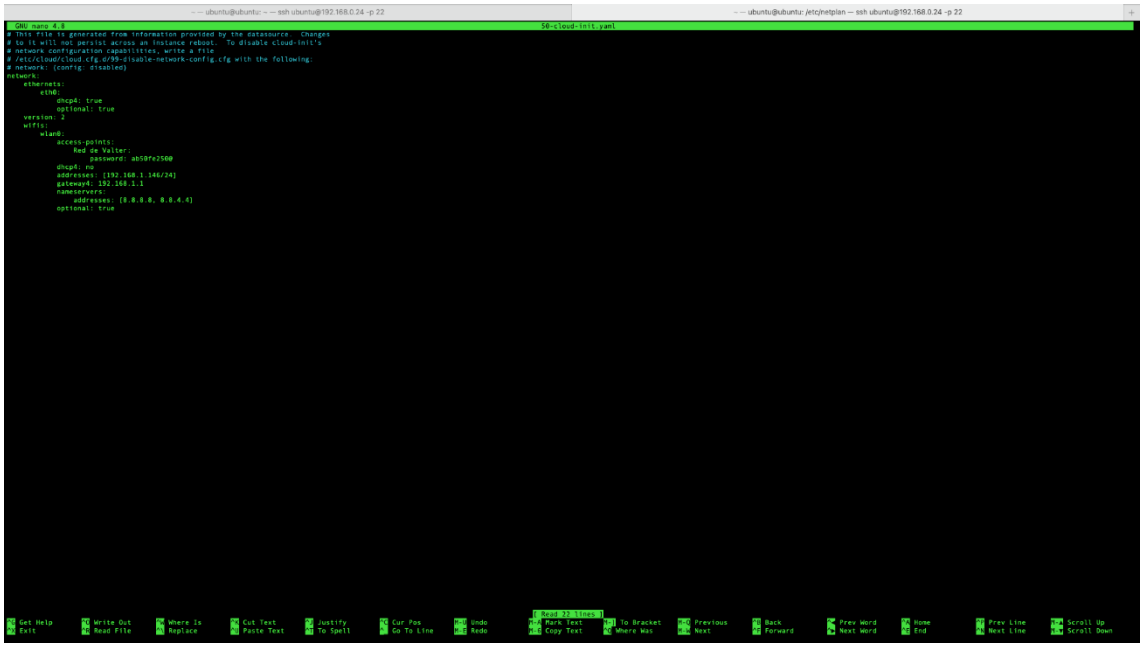
ssh ubuntu@192.168.0.24 -p 22

El usuario y la contraseña por defecto es “ubuntu”, aunque en el primer inicio nos solicitaran cambiar la contraseña, la cual hemos cambiado por “openhabian”.

Primero asignaremos una ip estática para la interfaz inalámbrica. Para ello ejecutaremos los siguientes comandos:

ubuntu@ubuntu:~$ cd ..
ubuntu@ubuntu:/home$ cd ..
ubuntu@ubuntu:/$ cd etc
ubuntu@ubuntu:/etc$ cd netplan/
ubuntu@ubuntu:/etc/netplan$ sudo nano 50-cloud-init.yaml

y dejaremos el archivo como la imagen mostrada a continuación:



De esta manera nos aseguramos de que siempre tenga la misma IP y poder configurarla de manera que no nos toque buscar cada vez la dirección IP.

Seguidamente, instalaremos el servidor MQTT Mosquitto, para que reciba los comandos MQTT que enviara el Arduino. Utilizaremos los siguientes comandos para instalarlo:

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-buster.list
sudo apt-get update
sudo apt-cache mosquitto
sudo apt-get install mosquitto
sudo systemctl enable mosquitto.service

Estos comandos los encontraremos en la documentación oficial de Mosquitto [9].

Con esto, ya tendríamos nuestra Raspberry Pi instalada con un sistema operativo, y configurada para empezar a trabajar con ella. Hay más opciones más fáciles para realizar nuestro proyecto, como por ejemplo una distribución llamada openhabian, la cual ya viene con OpenHAB ya instalado y solo haría falta configurarlo. Dado que nosotros tenemos más planes en mente, hemos decidido utilizar un Ubuntu Server LTS para que una persona al leer este proyecto pueda utilizar la raspberry pi para más cosas no relacionadas con el mundo de IoT.

6.2. Instalación y configuración de openHAB

Ya instalado el SO y el servidor MQTT que utilizaremos en el proyecto, continuaremos con la instalación y la configuración de openHAB. Para ello primero deberemos descargar desde el repositorio oficial la versión que vamos a utilizar de OpenHAB. Elegiremos qué interfaz vamos a utilizar de las tres que nos ofrece el propio sistema y describiremos cada uno de ellos.

Para empezar, instalaremos openHAB siguiendo la documentación de la página oficial de openHAB. [10]

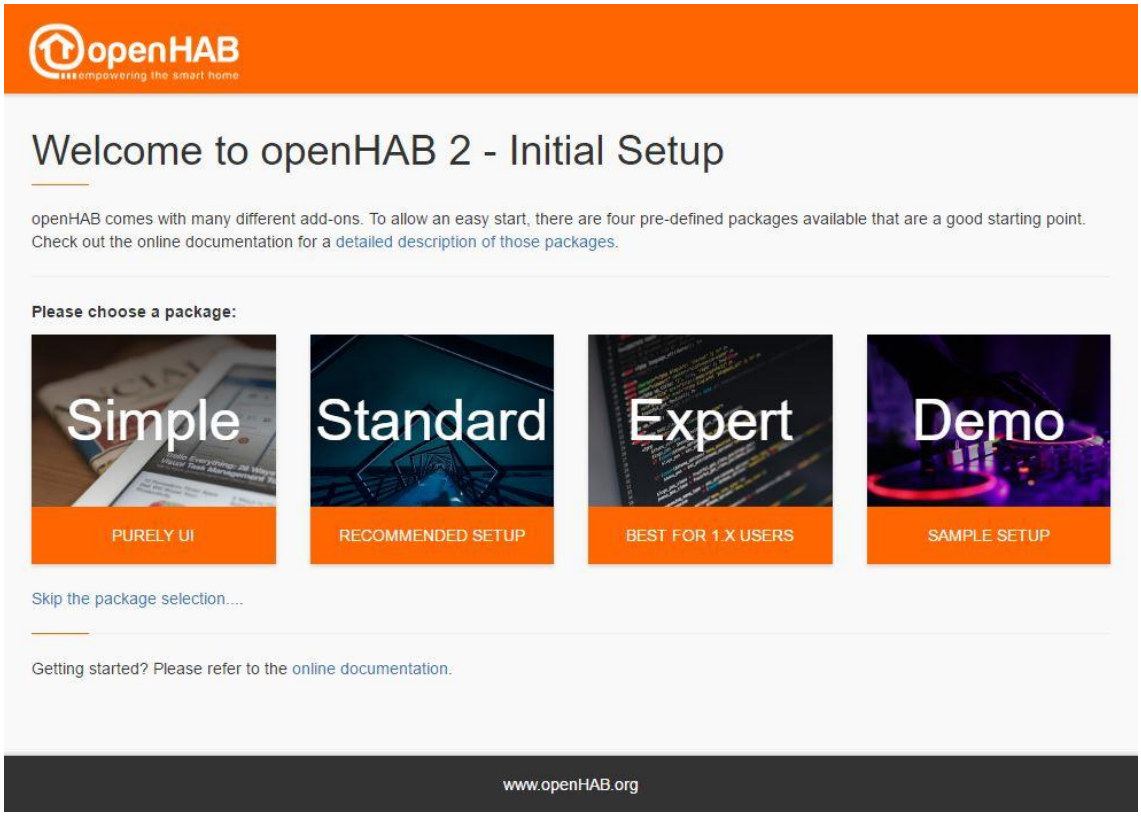
No entraremos a describir que hace cada uno de estos comandos, ya que están descritos en la documentación, aunque sí que aclararemos por qué estos comandos y no otros. en la documentación oficial encontramos varios tipos de instalación diferentes aun siendo el mismo sistema al que vamos a instalar. Esto se dependerá del tipo de sistema que tengamos instalado. en nuestro caso tenemos un Ubuntu Server que utiliza el sistema de archivos Apt Based.

Ejecutaremos los siguientes comandos:

```
wget -qO -'https://openhab.jfrog.io/artifactory/api/gpg/key/public' | sudo apt-key add -
sudo apt-get install apt-transport-https

echo 'deb https://openhab.jfrog.io/artifactory/openhab-linuxpkg stable main' | sudo tee /etc/apt/
sources.list.d/openhab2.list

sudo apt-get update

sudo apt-get install default-jdk

sudo apt-get install default-jre

sudo apt-get install openhab2

sudo apt-get install openhab2-addons

sudo systemctl start openhab2.service

sudo systemctl status openhab2.service

sudo systemctl daemon-reload

sudo systemctl enable openhab2.service
```

Después de unos 15 minutos debería estar todo instalado, dependiendo de la velocidad de nuestra conexión.

Al acabar, procederemos a ingresar mediante el navegador web de otro pc a la dirección IP de la raspberry y al puerto 8080. Introduciremos en nuestro caso lo siguiente: 192.168.1.146:8080

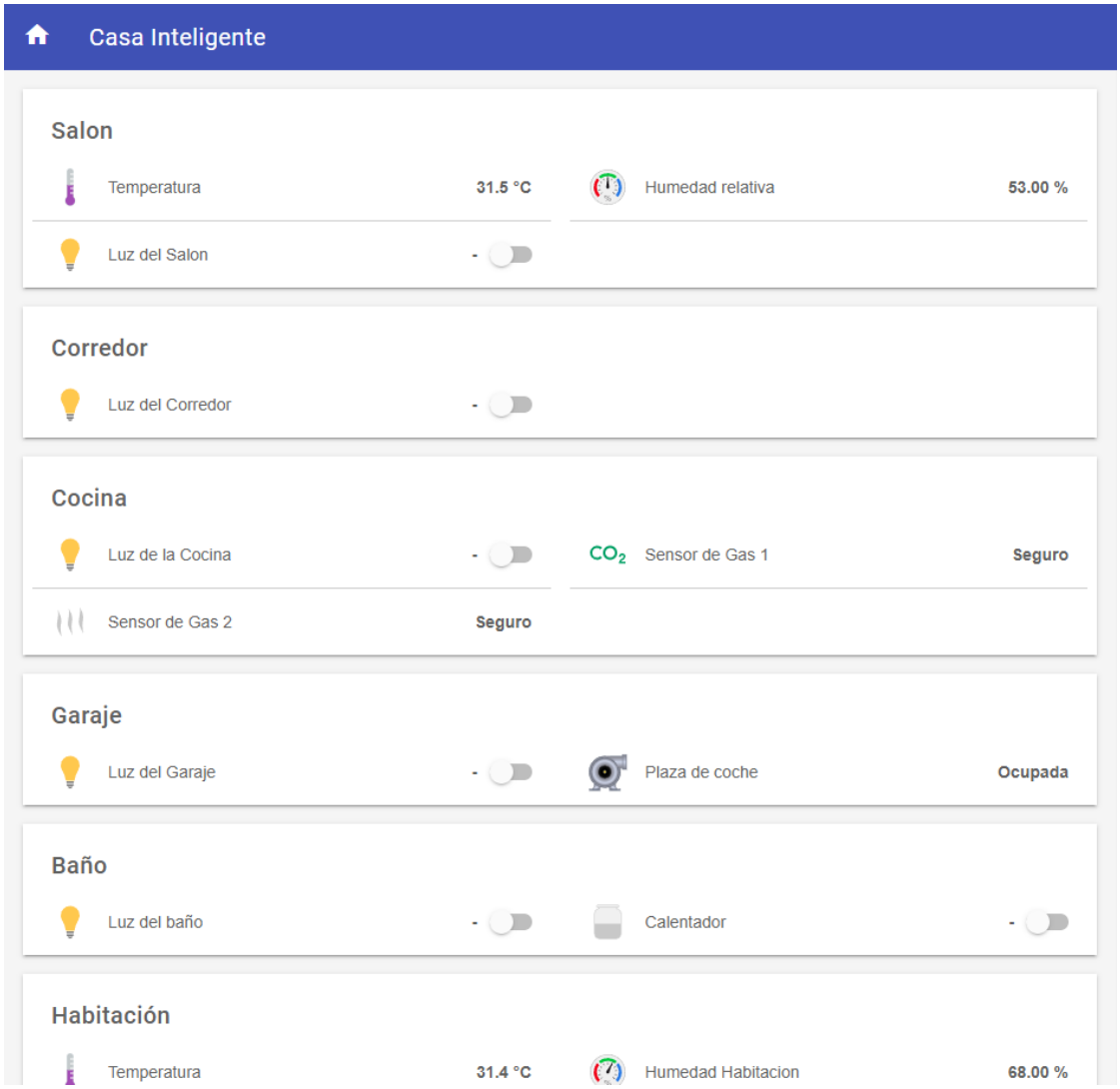


Nos aparecerá la imagen que tenemos arriba, en la cual nos dejará elegir entre 3 tipos:

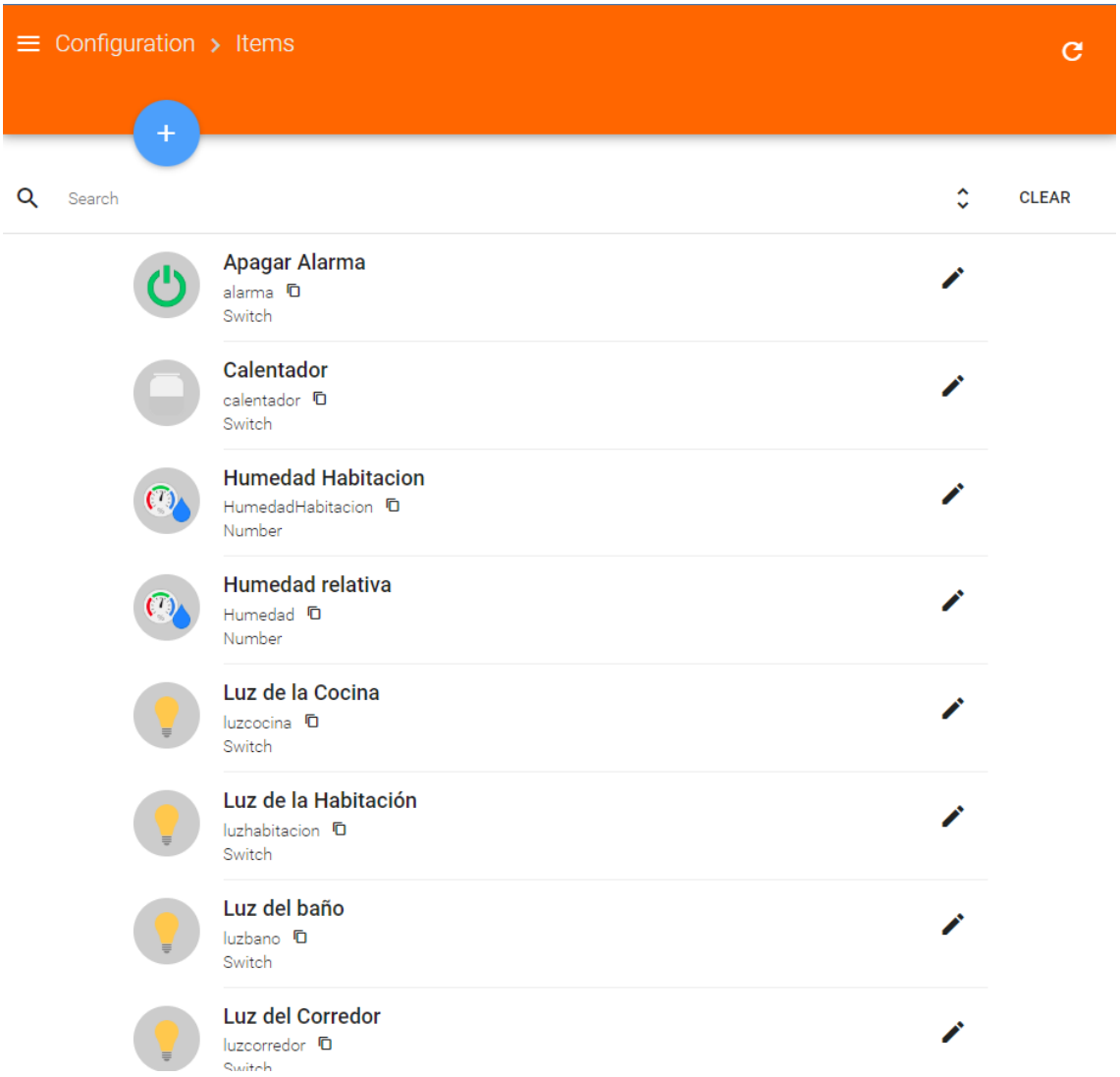
- Simple: Todo se controlará desde aquí. Se añadirán sensores y configurara todo desde esta interfaz gráfica. Esta opción sería la ideal si no se quiere configurar archivos o no se tienen nociones de programación.
- Standard: Esta sería la recomendada para el 90% de los usuarios. incluiría Basic UI, Habpanel y Paper UI. En este caso sí que será necesario configurar ciertos aspectos desde los archivos de configuración.
- Expert: Esta opción es la mejor si se está transaccionando desde OpenHab 1.x a OpenHab 2.x. Incluirá, además de lo anteriormente nombrado, Habadmin y Rest Api.

Nosotros elegiremos esta última para tener acceso completo a OpenHab, por si queremos en un futuro continuar con un proyecto más ambicioso.
En este punto ya instalado el sistema y ejecutado el setup inicial, después de elegir el modo experto nos quedarán en pantalla seis opciones. De momento utilizaremos tres de ellas, que describiremos a continuación:

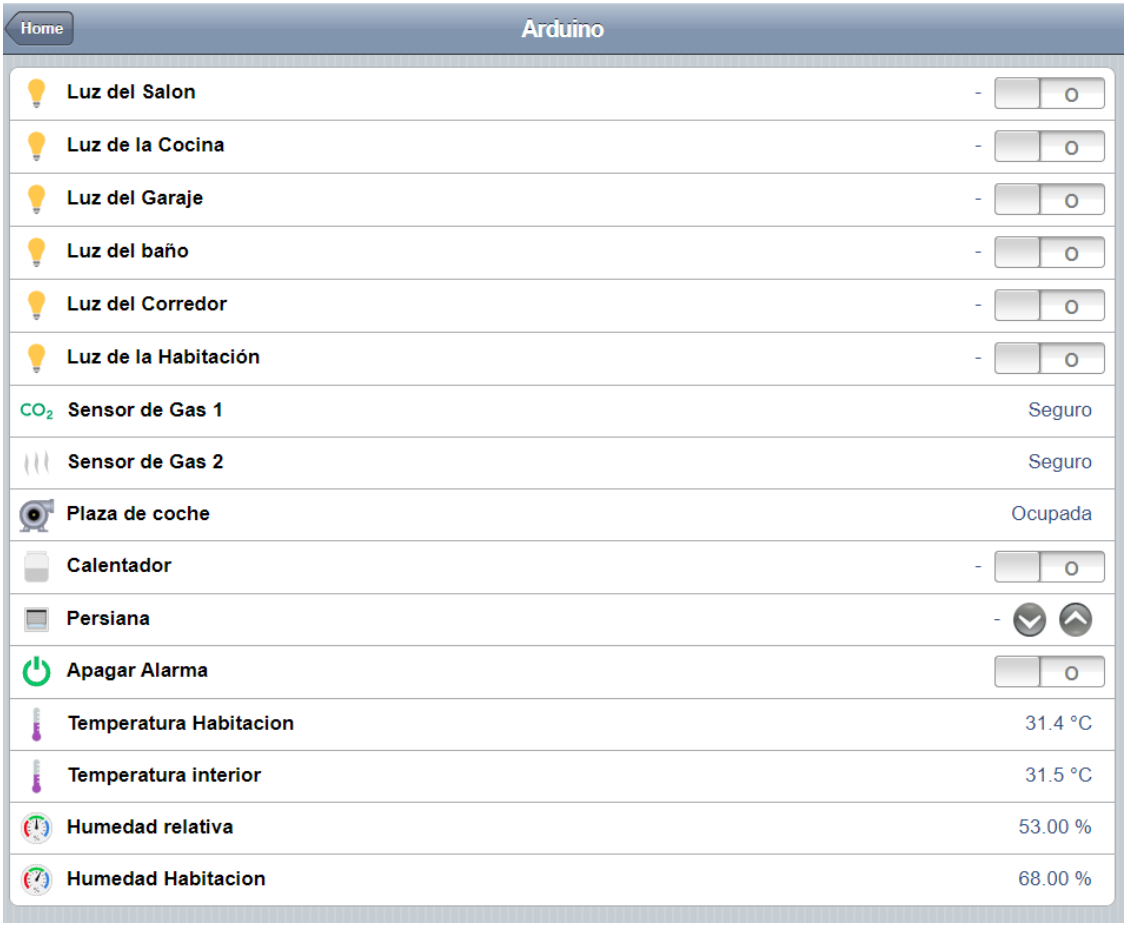
- Basic UI: esta opción será un panel de control donde podremos ver los diferentes sitemaps que tengamos creados. esta será la parte visual que veremos tanto desde nuestro ordenador como la aplicación móvil. podemos tener varios sitemaps creados con diferente información en cada uno.



- Paper UI: OpenHAB nos proporciona esta interfaz gráfica para instalar a nuestro servidor funciones de manera automática y sin necesidad de conocimientos avanzados de programación. este método está pensado para usuarios principiantes ya que les permite descubrir los “things”, indicarles a que “ítems” deben buscar para conseguir los datos que proporcionan y asociarlos a los “channels” correspondientes.



- Classic UI: desde este apartado veremos la misma información que en BASIC UI con la diferencia de qué está pensado visualmente para ser visto desde un terminal móvil. tiene la apariencia de los antiguos menús de opciones que Apple implementó en sus terminales.



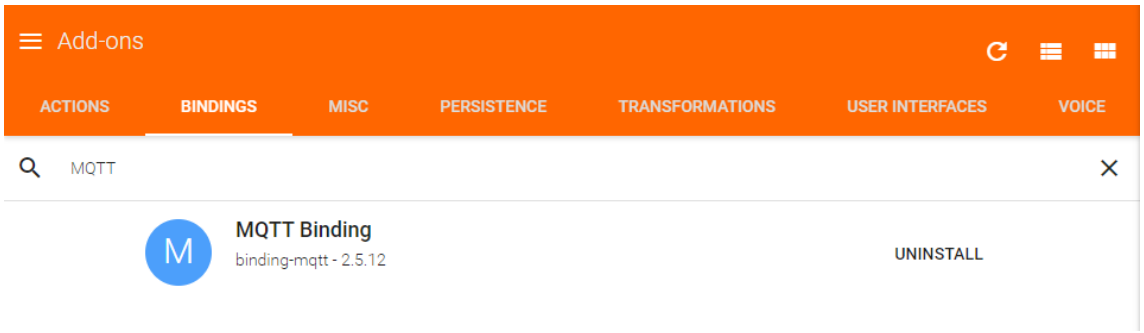
6.3. Instalación de Mycroft.

Visitando la documentación oficial de Mycroft [11] podemos ver una tabla con diferente hardware testado que funciona perfectamente. En nuestro caso, tenemos un micrófono Razer Seiren X, que no está soportado oficialmente. Antes de instalar todo nuestro proyecto, hicimos varias pruebas con Mycroft y nuestro micro y no pudimos hacerlo funcionar correctamente, ya que Mycroft si reconocía los comandos escritos, pero no detectaba sonido alguno. El sistema operativo no tenía soporte para ese micrófono, con lo que no podíamos hacerlo funcionar de manera correcta y decidimos dejar de lado esta idea y buscar alternativas. Para esta parte del proyecto, decidimos usar la inteligencia artificial Alexa.

6.4. Configuración OpenHAB

En este apartado iremos realizando diferentes acciones tanto en Paper UI como desde Terminal, ya que tenemos que configurar algunos archivos primero para poder continuar con la configuración.

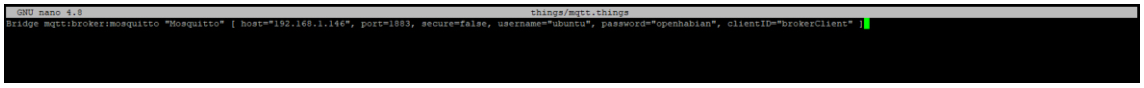
Para empezar, nos dirigiremos a Configuración, dentro de Paper UI, y seguidamente a Bindings. En este paso instalaremos MQTT Binding, que nos permitirá configurar conexiones a los brókers desde las things de openHAB. Dentro de Bindings, pulsaremos el botón +, e iremos a la sección Bindings, donde buscaremos en la barra de búsqueda MQTT Binding como vemos en la imagen a continuación:



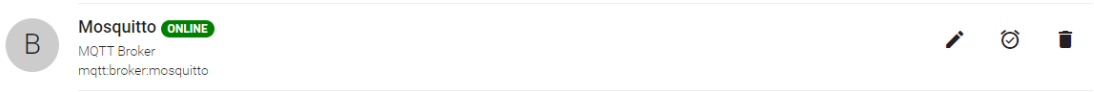
Una vez instalado, iremos a terminal y nos desplazaremos hasta la carpeta “etc/openhab2/things” Y crearemos un nuevo archivo, al que nosotros llamaremos “mqtt.things”. Es importante que acabe con esta extensión ya que OpenHAB leerá este archivo y lo añadirá dentro de Paper UI en la sección correspondiente.

En nuestro caso vamos a crear un Bridge para las conexiones MQTT entre Arduino y openHAB. para ello escribiremos lo siguiente dentro del archivo:

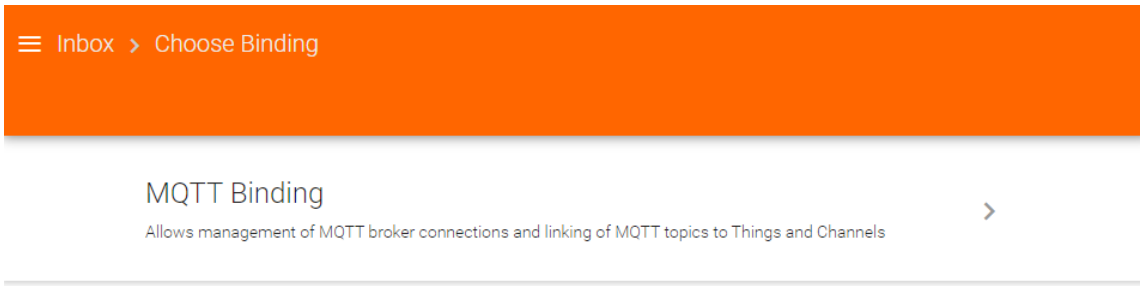
Bridge mqtt:broker:mosquitto "Mosquitto" [host="192.168.1.146", port=1883, secure=false, username="ubuntu", password="openhabian", clientID="brokerClient"]



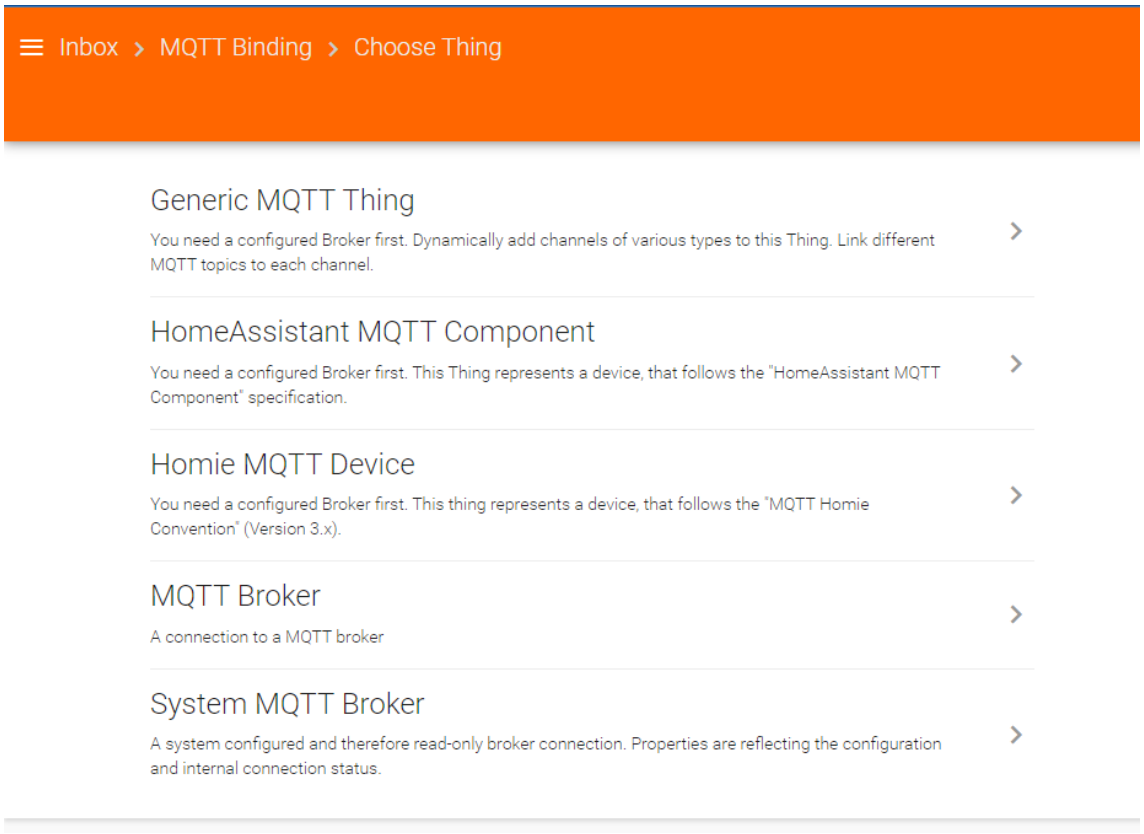
Esta instrucción describe los parámetros necesarios para realizar la conexión con el servidor mosquito, cómo puede ser la IP, usuario y contraseña de conexión, etc. guardaremos el archivo y podremos observar cómo en Paper UI, dentro de configuración en la pestaña Things, encontraremos este MQTT broker.



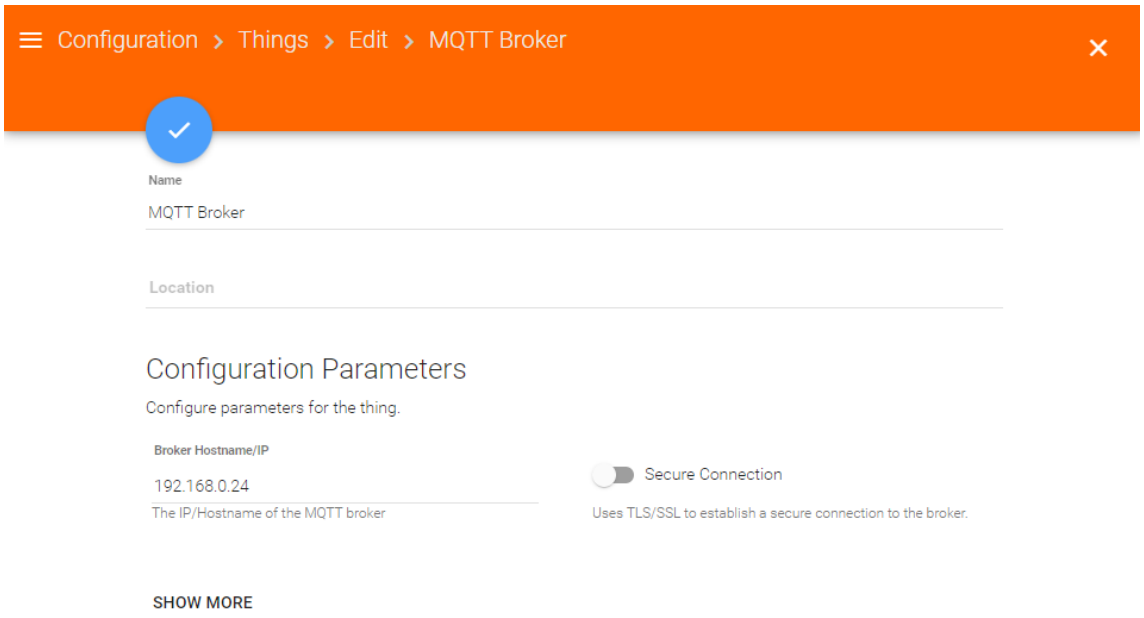
En nuestro caso, como tenemos dos conexiones diferentes también vamos a crear otro bróker desde esta misma página. Para ello pulsaremos el botón + qué encontramos en esa misma página y elegiremos la única opción disponible.



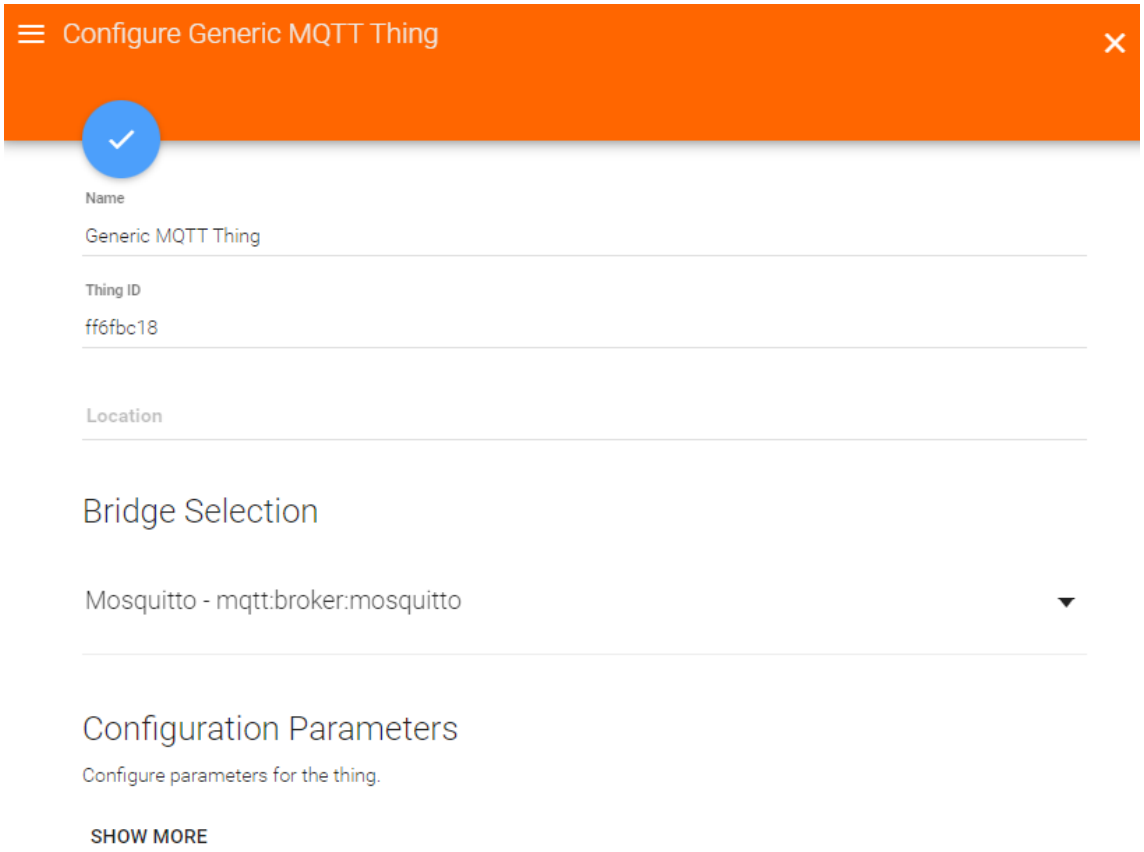
En la pantalla resultante pulsaremos para añadir un Thing manualmente, y elegiremos MQTT Bróker.



En la siguiente página nos encontraremos que nos piden los mismos parámetros que hemos introducido en el archivo mqtt.things, pero esta vez de manera gráfica. introduciremos la IP del servidor MQTT y les daremos a validar en la V azul.

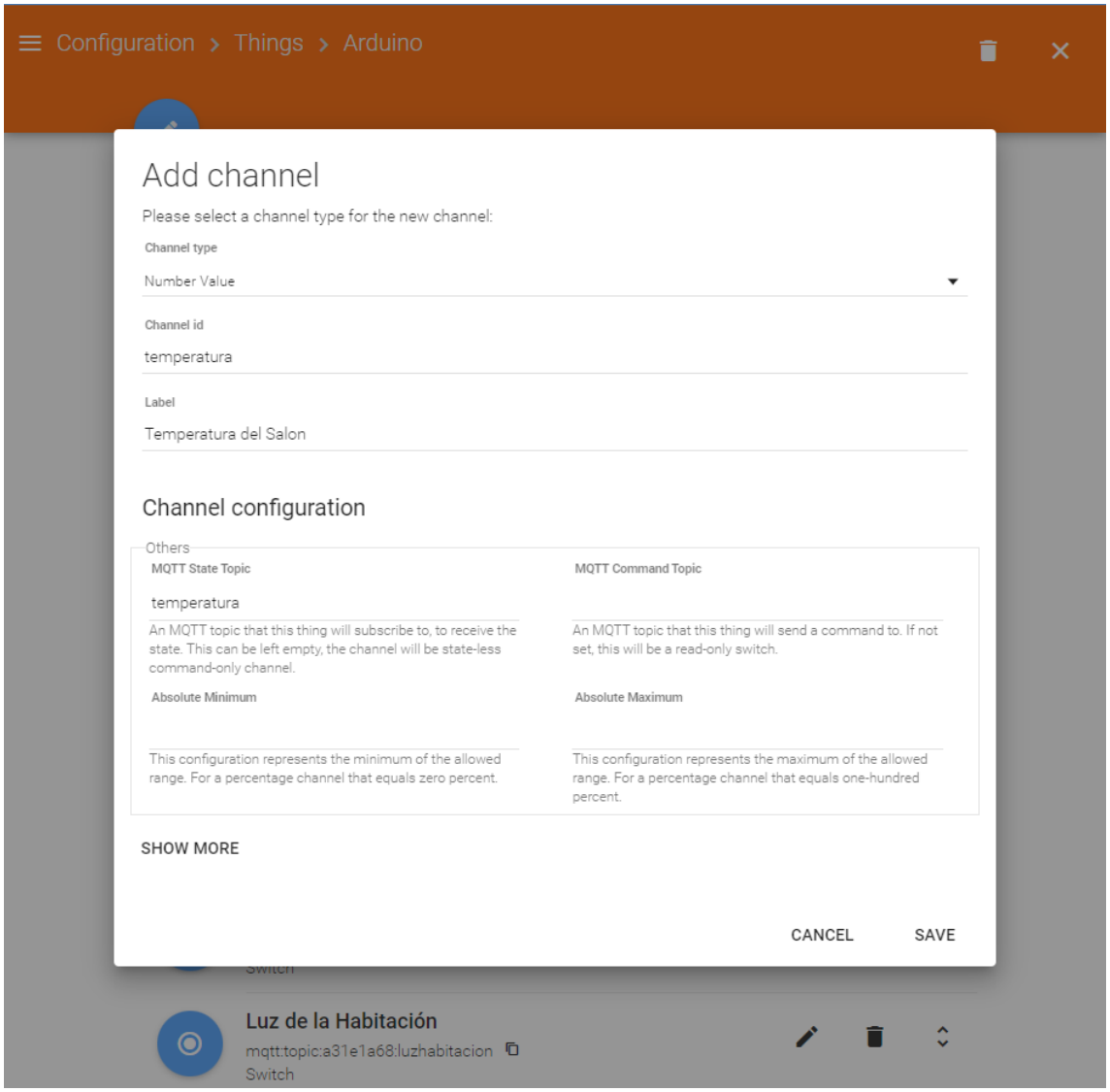


De esta manera, veremos cómo se nos va a hacer más fácil trabajar con dos redes diferentes, ya que el siguiente paso es crear un thing genérico donde tendremos los channels y sensores. Para ello, haremos los mismos pasos que con el bridge, pero esta vez elegiremos Generic MQTT Thing, y en bridge selection elegiremos el bridge configurado para la conexión en la que estemos.



En este caso le hemos cambiado el nombre y se llamara Arduino, ya que en el tendremos todo aquel ítem que se relacione con la placa de Arduino.

Hecho ya el thing, crearemos los channel que utilizaremos para los ítems que crearemos después. Para ello, dentro del thing, pulsaremos el símbolo + que está al lado de Channels, seleccionaremos el tipo de dato que vamos a utilizar en el item, el Chanel id, el nombre con el que localizaremos el channel, y en MQTT State Topic introduciremos el canal de donde leerá la información proporcionada por el sensor, mientras que MQTT Command Topic será el canal al que openHAB le enviara un comando al sensor.



Hemos creado en este caso el channel para el sensor de temperatura ubicado en el Salón, pero crearemos todos los channels necesarios para todos los sensores. Mostramos solo este porque todos son iguales, cambiando los datos de configuración.

A continuación, añadiremos los ítems a openHAB mediante archivos de configuración.

Para ello, nos desplazaremos hasta etc/openhab2/ítems y crearemos un archivo para añadir los ítems. Con el siguiente comando dentro de esta carpeta crearemos el archivo:

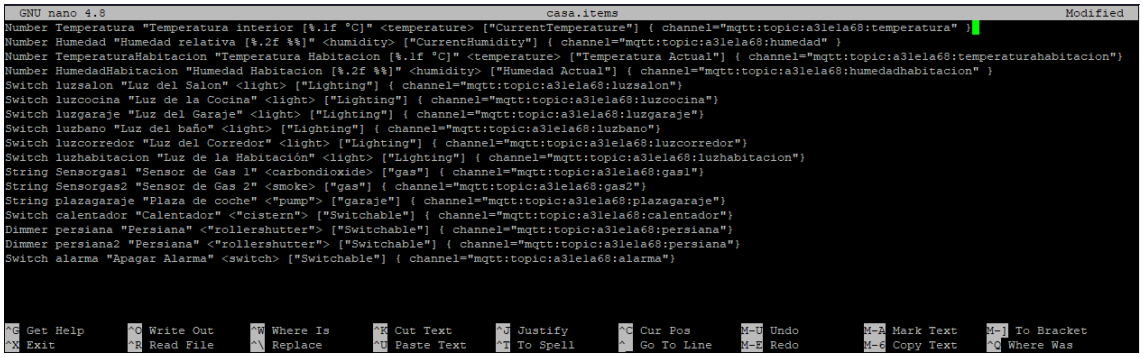
$ sudo nano casa.items

A partir de aquí podremos añadir los ítems según el siguiente esquema:

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

Como podemos ver, definiremos el tipo de dato que será el ítem, el nombre, el nombre que tendrá el ítem en Paper UI y en Basic UI, el formato que tendrá el dato mostrado, el icono que tendrá en Basic UI, el tag que utilizara si queremos usarlo con otro bind, y el bind que utilizara el ítem. En nuestro caso mostraremos uno de los ítems, aunque podremos ver en la imagen como quedara nuestro archivo:

Number Temperatura "Temperatura interior [%.1f °C]" <temperature> ["CurrentTemperature"] { channel="mqtt:topic:a31e1a68:temperatura" }



Aquí hemos definido cada uno de los ítems que usaremos en el proyecto. Podemos ver que hemos utilizado como channel el que hemos creado antes, que nos ha proporcionado parte de esa dirección y hemos añadido el canal donde se publicara la información por parte de la placa de Arduino.

Continuaremos con la configuración del sitemap. El sitemap es la parte más importante para poder visualizar los datos de manera ordenada y detallada en el Basic UI. Para ello, como en la ocasión anterior, deberemos movernos a la carpeta etc/openhab2/sitemap.

En esta carpeta crearemos con el siguiente comando el sitemap que utilizaremos en nuestro proyecto:

$ sudo nano casa.sitemap

La sintaxis de nuestro sitemap según la documentación oficial de openHAB sería la siguiente:

sitemap <sitemapname> label="<title of the main screen>" {
  ItemType = ItemName label = “Decription on the item”
}

Siguiendo esta sintaxis, y continuando con el ejemplo anterior, nos encontraremos lo siguiente en nuestro archivo de configuración:

sitemap demo label="Casa Inteligente" {
  Frame label="Salon" {

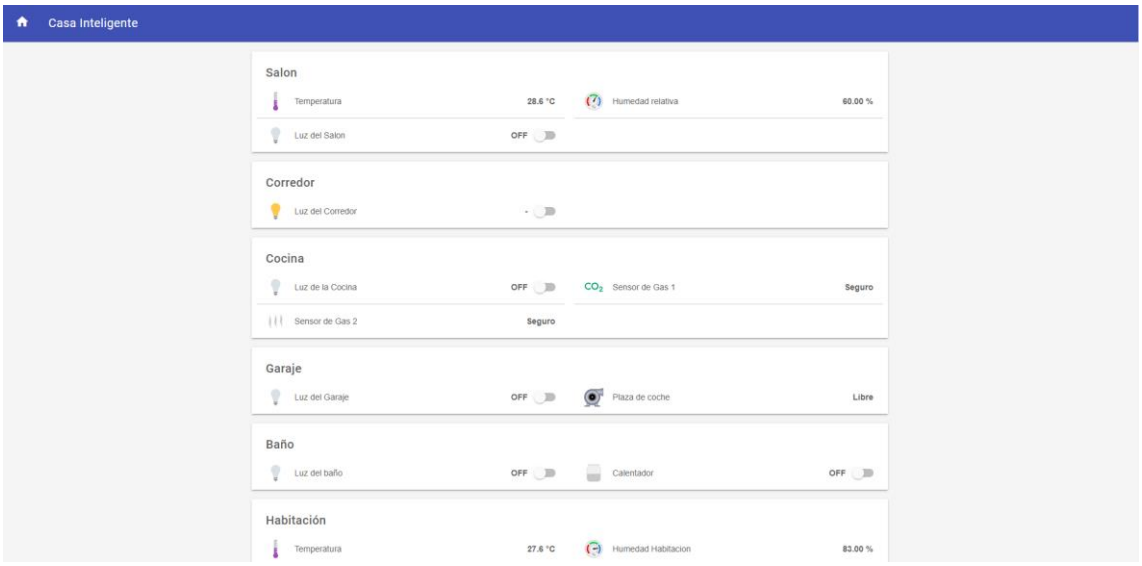
```
    Text item=Temperatura label="Temperatura [%.1f °C]"
}
```

Con ello conseguiremos que el título de la página sea Casa Inteligente, y que dentro de la página, este una división con título Salón y dentro de esta división tendremos el ítem de temperatura.

A continuación, veremos cómo nos queda distribuido el sitemap en nuestro proyecto con todos los ítems subdivididos en sus partes de la casa correspondientes.

```
  GNU nano 4.8                        casa.sitemap
sitemap demo label="Casa Inteligente" {
    Frame label="Salon" {
        Text item=Temperatura label="Temperatura [%.1f °C]"
        Text item=Humedad
        Switch item=luzsalon icon="light"
    }
    Frame label="Corredor" {
        Switch item=luzcorredor icon="light"
    }
    Frame label="Cocina" {
        Switch item=luzcocina icon="light"
        Text item=Sensorgas1 icon="carbondioxide"
        Text item=Sensorgas2 icon="smoke"
    }
    Frame label="Garaje" {
        Switch item=luzgaraje icon="light"
        Text item=plazagaraje icon="pump"
    }
    Frame label="Baño" {
        Switch item=luzbano icon="light"
        Switch item=calentador icon="cistern"
    }
    Frame label="Habitación" {
        Text item=TemperaturaHabitacion label="Temperatura [%.1f °C]"
        Text item=HumedadHabitacion
        Switch item=luzhabitacion icon="light"
        Switch item=persiana icon="blinds" mappings=[0="Cerrar", 100="Abrir"]
        Slider item=persiana icon="blinds"
    }
    Frame label="Notificaciones" {
        Switch item=alarma
}
}
```

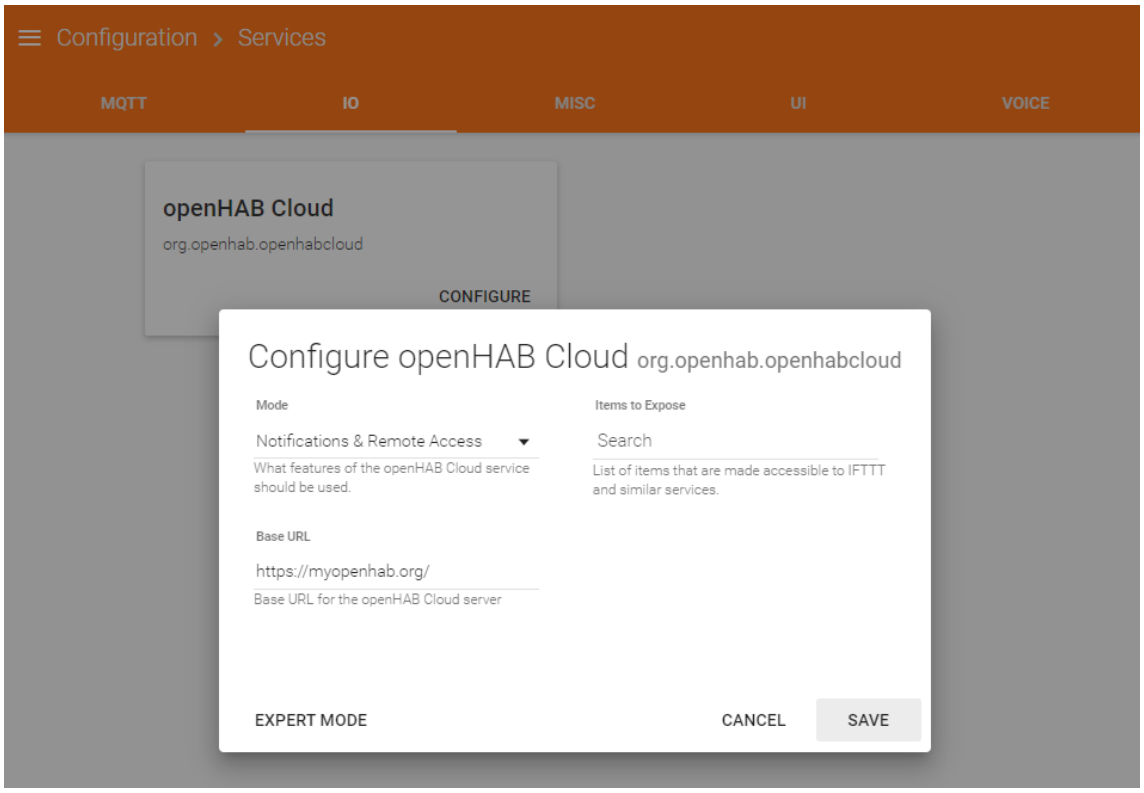
Después de guardar el archivo, si accedemos a Basic UI nos encontraremos con cada una de las divisiones como veremos en la imagen final de cómo queda la página web.



En este punto y para continuar con las rules, antes deberemos instalar cierto binding, pero instalaremos todos los bindings ya y los configuraremos. Instalaremos los bindings MQTT Bindings, que ya lo hemos hecho, OpenHAB Cloud Connector, Mycroft AI Skill, y llegado el momento Amazon Alexa Smart Home Skill, pero explicaremos su instalación en este punto.
Empezaremos por OpenHAB Cloud Connector.

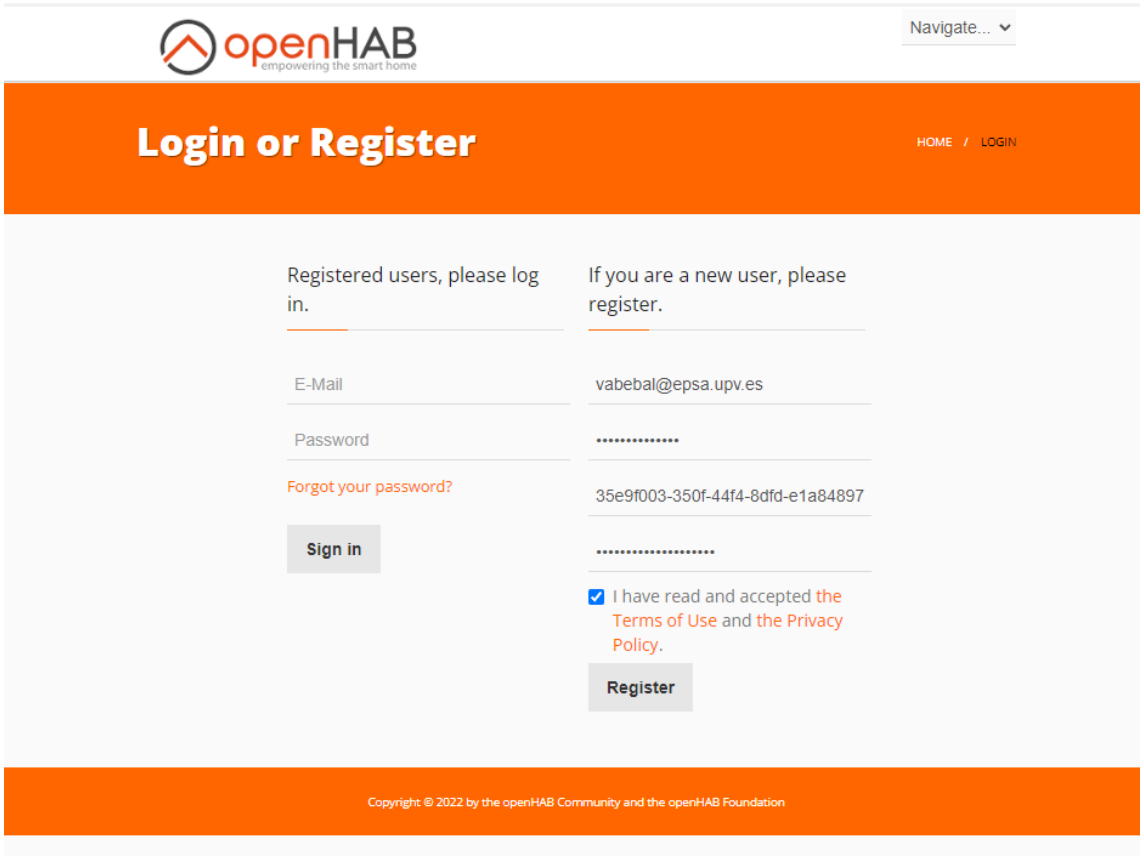
Esta integración nos da ciertas ventajas como permitir acceso remoto a instancias locales de openHAB sin tener que exponer puertos a Internet o tener la necesidad de instalar y configurar una VPN, Servir de conector entre Google Cloud Messaging o Apple Push notifications para publicar notificaciones en las apps mobiles y darnos posibilidades de integración con servicios que requieren de autenticación OAuth2.

Para ello, como nos indica la documentación oficial de openHAB, deberemos entrar en nuestro portal web de openHAB, entraremos en Paper UI y buscaremos la sección de Add-ons. Dentro de esta sección, entraremos en Misc y buscaremos openHAB Cloud Connector. Le daremos a instalar.
Seguidamente, nos dirigiremos a Configuration y a la sección Services y ya nos aparecerá openHAB Cloud. Entraremos a configurarlo y en Mode indicaremos la opción Notifications & Remote Access, y en Base URL escribiremos: https://myopenhab.org/



Para poder acceder necesitaremos registrarnos, para ello necesitaremos la información de dos archivos del sistema, así que para conseguir dicha información nos dirigiremos primero a /var/lib/openhab2 donde encontraremos primero el uuid. Usando cat uuid nos mostrara el contenido del uuid que deberemos anotar para registrarnos posteriormente. En nuestro caso, 35e9f003-350f-44f4-8dfd-e1a84897cd51.

A continuación, nos dirigiremos a /var/lib/openhab2/openhabcloud/ donde encontraremos el archivo secret, y haciendo la misma operación que antes, nos dará como resultado el secret que tenemos que utilizar siendo en nuestro caso yUrmzUn1hnDRpBMoebdw.
Con esta nueva información, nos dirigiremos a https://myopenhab.org/login y nos registraremos con los datos necesarios.



Una vez dentro, podremos acceder de manera remota a nuestro Hub de OpenHAB.

Para nuestro proyecto no usaremos reglas o rules, pero si programaremos una para poder demostrar el uso de las rules y su utilidad, dándole una posible continuación al proyecto. Las rules son reglas de comportamiento que, según ciertas condiciones, provocaran una respuesta en el sistema. Pueden cambiar el estado de los sensores dependiendo de las condiciones que les hayamos programado a dichas reglas.

Como ya hemos programado un interruptor que no tiene utilidad en la placa del Arduino, utilizaremos ese mismo interruptor para crear un aviso si salta la alarma. Para ello nos desplazaremos a la carpeta /etc/openhab2/rules/.

Según la documentación oficial, las reglas tendrán la siguiente sintaxis:

```
rule "<RULE_NAME>"
when
  <TRIGGER_CONDITION> [or <TRIGGER_CONDITION2> [or ...]]
then
  <SCRIPT_BLOCK>
end
```

Contará con un nombre, una o varias condiciones y el resultado que se desarrollara si estas condiciones se cumplen. Nosotros solo queremos mostrar su funcionamiento, con lo que haremos una regla simple para mandar una notificación al teléfono en caso de que se active ese interruptor.

Para ello crearemos dentro de la carpeta rules un archivo llamado notifications.rules.

En el analizaremos si el ítem alarma que es un interruptor, cambia de estado, entonces enviara una notificación al móvil que tenga instalada y configurada la aplicación de openHAB. Utilizaremos la instrucción sendNotification, donde tendremos que introducir el correo electrónico con el que nos hemos registrado en openHAB Cloud seguido del mensaje que queremos que aparezca en la notificación push de nuestro teléfono. En la imagen siguiente podemos ver como quedaría esta regla.

```
  GNU nano 4.8                    notifications.rules
rule "Send Mobile Push Notification"
when
    Item alarma changed
then
    if(alarma.state == ON)
    {
        logInfo("notifications", "Sending notification via app.")
        sendNotification("vabebal@epsa.upv.es",
            "Se ha disparado la alarma. Verifique y apague la sirena")

        alarma.postUpdate(OFF)
    }
end
```

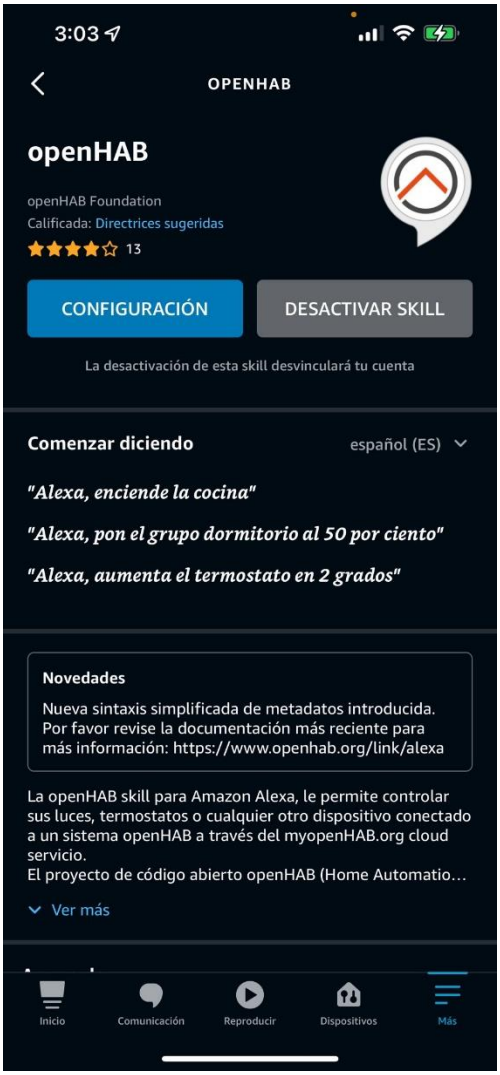
En esta imagen podemos ver la notificación que se mostrara, cada vez que activemos dicho interruptor. Dejaremos esta opción como posible mejora del sistema, que será muy fácil de implementar. Tan solo deberíamos hacer que Arduino publique en un canal MQTT propio de la alarma y haríamos que cambiase el estado de dicho interruptor. Al cambiar el estado, nos enviaría una notificación al teléfono, con el mensaje de la imagen.

6.5. Otras instalaciones y configuraciones.

Como hemos dicho anteriormente, hay más cosas a configurar como por ejemplo Alexa, el IDE de Arduino y la aplicación móvil de openHAB.

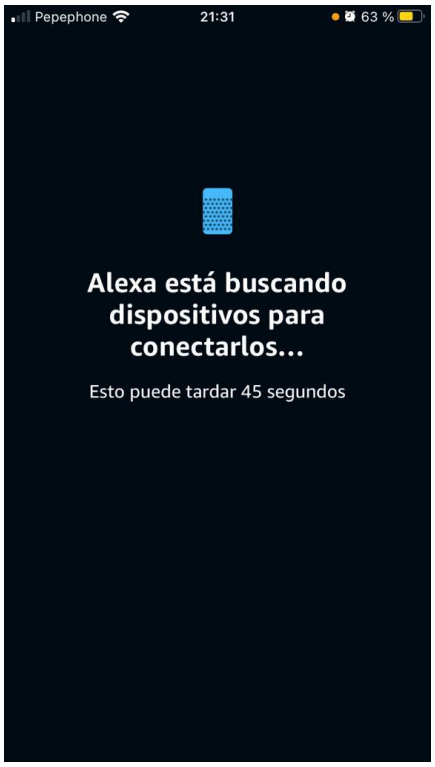
6.5.1. Configuración de Alexa.

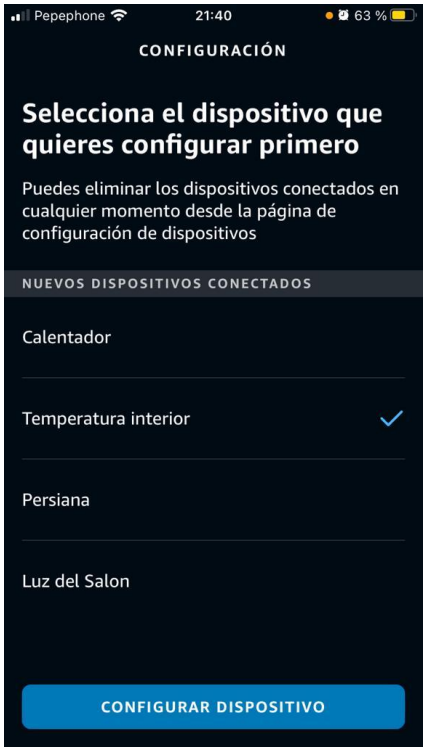
Empezaremos por Alexa. Instalaremos la aplicación Amazon Alexa en nuestro terminal, ya que así podremos desde cualquier parte mientras tengamos conexión a internet, tener acceso por voz a openHAB. Al instalarla, y registrarnos con nuestro usuario y contraseña de Amazon. Una vez dentro nos dirigiremos a la parte inferior de la pantalla y entraremos en Más, dirigiéndonos posteriormente a Skills y Juegos. Usaremos la lupa en la parte superior de la aplicación y buscaremos openHAB.



La instalaremos y cuando acabe de configurarse nos aparecerá las siguientes ventanas, dándonos a elegir que dispositivos queremos configurar primero:


Pepephone 21:31 63 %
VINCULACIÓN DE CU...
Tu cuenta de openHAB se ha vinculado correctamente.
Siguientes pasos:
Cierra esta ventana para detectar dispositivos de Hogar digital que puedas controlar con Alexa.
CERRAR


Pepephone 21:31 63 %
Alexa está buscando dispositivos para conectarlos...
Esto puede tardar 45 segundos


Pepephone 21:40 63 %
CONFIGURACIÓN
Selecciona el dispositivo que quieres configurar primero
Puedes eliminar los dispositivos conectados en cualquier momento desde la página de configuración de dispositivos
NUEVOS DISPOSITIVOS CONECTADOS
Calentador
Temperatura interior
Persiana
Luz del Salon
CONFIGURAR DISPOSITIVO


Pepephone 21:41 63 %
CONFIGURAR
Se ha añadido Temperatura interior a Dormitorio
Ya puedes controlar este dispositivo con el nombre de la estancia y el tipo de dispositivo, no será necesario que recuerdes el nombre del dispositivo.
CONTINUAR

Cabe destacar que esta configuración se queda guardada en la nube, por lo que al conseguir otro dispositivo compatible con Alexa o que tenga Alexa integrado (Amazon Dot, una Smart tv…) y vincularlo con nuestra cuenta de Amazon, seremos capaces de controlar nuestro proyecto desde este nuevo dispositivo.

En este caso, mostramos como se ha llevado a cabo la configuración del sensor de temperatura. No mostraremos más ya que es exactamente igual con todos los que restan y prácticamente es automático.

6.5.2. Arduino IDE

Para programar Arduino, tenemos infinidad de IDEs diferentes que podemos utilizar como su propio IDE, Sublime text, Visual Studio… Nosotros nos centraremos en el IDE que Arduino pone a nuestra disposición de manera gratuita en su página web.

Después de descargarlo e instalarlo, abriremos el IDE y cambiaremos ciertas opciones que nos ayudaran en un futuro si ocurre algún error a la hora de subir el sketch a nuestra placa, ofreciendo más información. Para ello iremos a Archivo y nos desplazaremos hasta Preferencias. Marcaremos las casillas de compilación y subir en la sección “Mostrar salida detallada mientras:” y marcaremos también “Mostrar números de línea”.

También instalaremos las bibliotecas que utilizaremos en nuestro proyecto, como la biblioteca PubSubClient, o la biblioteca para el uso de LCD i2c de fmalpartida, entre otras. Algunas las podremos instalar desde el propio IDE. Para ello nos desplazaremos a Programa, después a Incluir librería y Administrar bibliotecas. En la barra de búsqueda, introduciremos la biblioteca deseada y le pulsaremos el botón de instalar. Nosotros instalaremos:

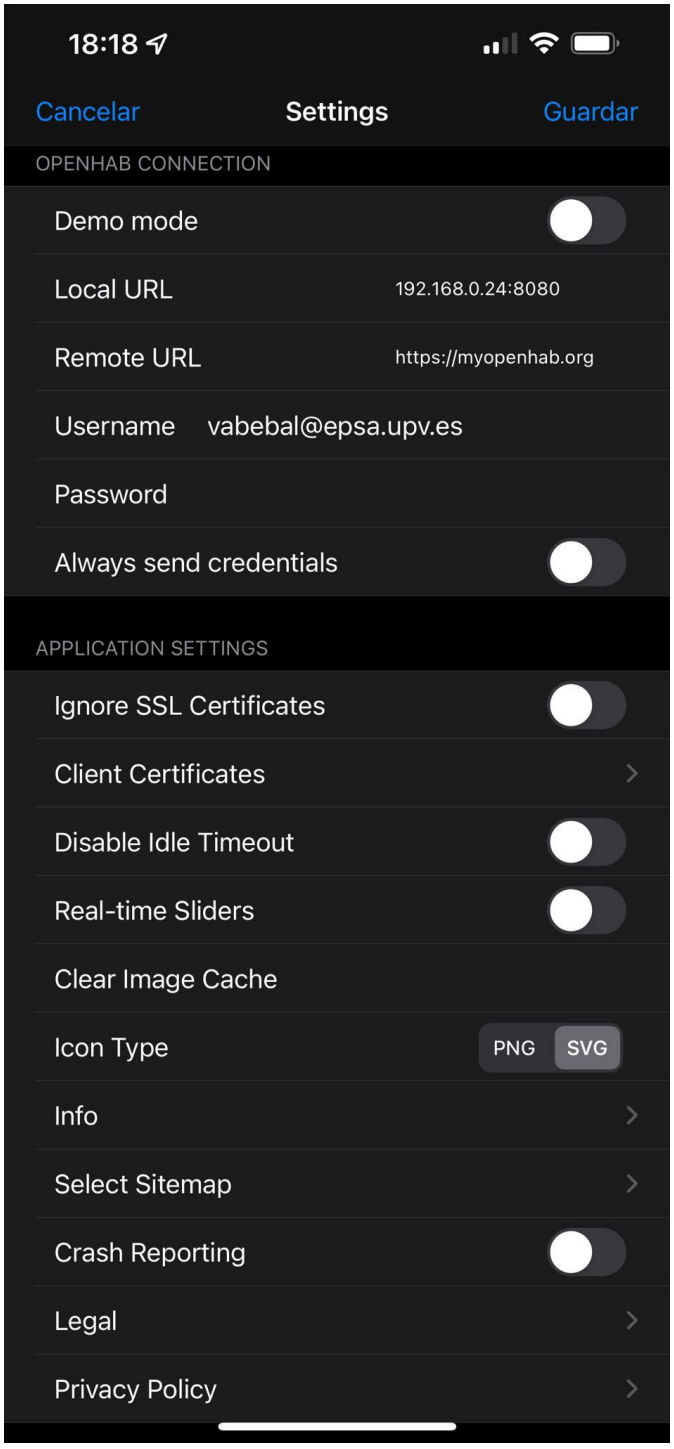
- DHT sensor library
- MFRC522
- PubSubClient

Además, también instalaremos otras librerías que descargaremos desde internet en formato .zip. De esta manera, instalaremos las siguientes bibliotecas:

- New-LiquidCrystal de fmalpartida desde el link de su github: https://github.com/fmalpartida/New-LiquidCrystal
- NewPing para controlar el sensor de ultrasonidos: https://bitbucket.org/teckel12/arduino-new-ping/downloads/NewPing_v1.9.4.zip
- Keypad para controlar una matriz 4x4: http://playground.arduino.cc/Code/Keypad
- Password para la alarma: http://playground.arduino.cc/Code/Password

Con esto ya tendremos instalado todo aquello que vamos a necesitar para empezar a trabajar con nuestras placas de Arduino.

Para acabar, instalaremos la aplicación de openHAB en los dispositivos móviles en los que queramos tener acceso. Abriremos la aplicación y nos dirigiremos a Settings. En este apartado ingresaremos cuatro datos necesarios: La IP local de la Rapberry PI incluido el puerto en el que esta alojado openHAB: 192.168.1.146:8080 , la URL remota: https://myopenhab.org , el usuario que hemos registrado en openHAB Cloud: vabebal@epsa.upv.es y la contraseña de este usuario. Nos quedaría esta pestaña como la imagen que tenemos a continuación:


18:18
Cancelar
Settings
Guardar
OPENHAB CONNECTION
Demo mode
Local URL
192.168.0.24:8080
Remote URL
https://myopenhab.org
Username
vabebal@epsa.upv.es
Password
Always send credentials
APPLICATION SETTINGS
Ignore SSL Certificates
Client Certificates
Disable Idle Timeout
Real-time Sliders
Clear Image Cache
Icon Type
PNG
SVG
Info
Select Sitemap
Crash Reporting
Legal
Privacy Policy

6.6. Arduino

Después de configurar el IDE y de descargar las librerías externas necesarias, procederemos a empezar a programar los diferentes sensores. Para ello nos podemos apoyar en los ejemplos que vienen con cada librería, entrando en el menú de Archivo, Ejemplos y buscando el sensor que queremos aprender a usar. Cada ejemplo tiene una explicación antes de empezar con la programación de dicho sensor. Por ejemplo, si entramos en uno de los ejemplos de la librería Keypad, nos encontraremos una explicación de su funcionamiento como podremos ver a continuación.

Gracias a esto y a nuestros conocimientos de programación, podremos ir haciendo pruebas para familiarizarnos con la programación de dicho sensor. En otros casos, deberemos recurrir a la documentación oficial la librería que queramos utilizar si no entendemos bien su funcionamiento o queremos saber las funciones de las que disponemos. Por ejemplo, la librería PubSubClient en sus ejemplos nos enseña cómo usarla, pero no nos da ni la sintaxis ni los tipos de argumento que debemos pasarle a las funciones para que funcionen. Podríamos querer hacer un “client.publish("outTopic","hello world");”[12] y queremos pasar un int en lugar de hello world. La función tendría esta sintaxis:

boolean publish (topic, payload, [length], [retained])

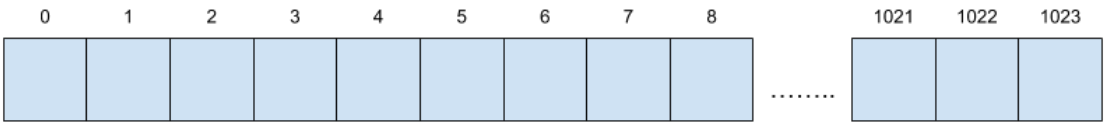
siendo los tipos de datos de topic const char[], de payload const char[] o byte[], de length unsigned int siendo opcional y retained un boolean.

Estos datos no nos los aporta el ejemplo, por lo que será importante recurrir a la documentación oficial para revisar estos aspectos y ver tanto las funciones de las que disponemos como de los tipos de datos que ofrecen, devuelven o tenemos que pasarle a dicha función.

Hemos recurrido a esta librería como ejemplo porque es las pocas que no conocíamos y que nos ha resultado realmente útil la documentación para la realización del proyecto. También es importante decir que las librerías utilizadas han sido ya vistas por el autor de este proyecto con anterioridad en otros proyectos, por eso sabia usar las otras librerías.

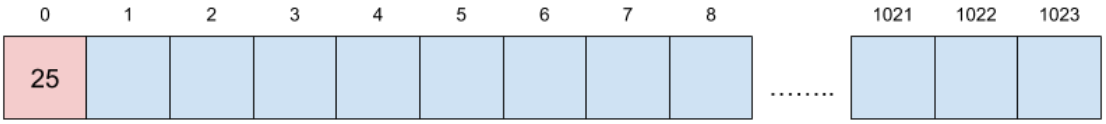
Otro aspecto importante por destacar es el uso de la librería EEPROM.h. Esta librería realiza operaciones en la memoria EEPROM (siglas en inglés de Electrically Erasable Programmable Read-Only Memory) de la placa de nuestro Arduino. Esta es una memoria no volátil, es decir, no pierde la información cuando nuestra placa pierde la alimentación, con una capacidad en las placas de Arduino MEGA de 4096 bytes. En nuestro caso, la utilizaremos únicamente para si se perdiese la electricidad, que openHAB sepa cuál es la apertura actual de las persianas y así que no haya ningún error de cálculo a la hora de subirlas o bajarlas. Es una memoria realmente que juega con el hándicap de tener un numero de usos limitado. La vida útil de esta memoria es de 100000 ciclos. Haremos un ejemplo para que quede más claro todo.

En un Arduino UNO tenemos de memoria EEPROM 1024 bytes. Vamos a colocarlos gráficamente para poder visualizar mejor este ejemplo. [13]



Si queremos guardar un byte de información, tendremos que indicar que queremos guardar y donde. Si queremos guardar el dato 25 en la posición 0, tendremos que indicarle al sistema dicha instrucción.

EEPROM.put(0, 25)



Como resultado se almacenará este dato en esa posición. Este hecho podemos realizarlo 100000 veces en esta posición. A partir de este número de ciclos, la memoria de Arduino no es consistente y puede provocarnos fallos. Podríamos cuando esta dirección de memoria sea inconsistente, cambiar la posición a la posición 1, y tendríamos otros 100000 usos. No sería difícil de programar con una instrucción con una instrucción que incremente en uno la variable posición cuando tenga 99999 ciclos. Tendríamos algo como el siguiente pedazo de código:

```
1  //Ejemplo ilustrativo de posible mejora de sistema
2  //Valentí Besó Ballester. 2022
3  int x = 0;
4  int y = 0;
5  Byte dato;
6
7  Void escribir(dato){
8  EEPROM.put(x, dato);
9  y++;
10 if (y = 99999){
11   x++;
12   y = 0;
13 }
14
```

Con esto mejoraríamos, considerablemente el diseño frente a fallos ya que, si subiésemos o bajásemos la persiana 2 veces al día, tardaríamos 204800000 días en tener toda la memoria EEPROM en el mismo estado de inconsistencia, es decir 561095 años en tener la memoria en un estado de inconsistencia. Habríamos desaprovechado 4096 ciclos salvando los datos de un "posible" estado de inconsistencia ya que Arduino da garantía de ese número de ciclos, aunque pueden ser más o menos.

Este tipo de memoria es útil para este tipo de diseños, pero se tiene que utilizar con precaución para no acabar en un bucle y escribir la memoria de manera reiterada y que acaba en estado de inconsistencia antes de tiempo. Por ello, hay que estar seguro de cómo estamos trabajando con ella.

Otra opción sería usar update cuando ya hay un dato escrito en una dirección, ya que esto podría salvar ciclos si los datos no cambiasen muchas veces. Aunque en nuestro caso, prácticamente no habrá diferencia.

También podríamos hablar un poco de la librería Ethernet.h. En nuestro proyecto ha sido una parte fundamental, ya que, sin comunicación con nuestra conexión a internet, no podíamos conseguir controlar la placa de Arduino desde openHAB. Esta librería ya había sido usada con anterioridad por el autor para poder crear un servidor web para pruebas de programación y ver el potencial que tenía el Shield de Ethernet.

Ciertos códigos como el lector RFID [14] han sido obtenidos desde internet y modificados en parte o en su totalidad para el correcto funcionamiento en nuestro proyecto, ya que los ejemplos que nos daban eran pobres en comparación con el resultado buscado.

Para tener el código ordenado hemos utilizado librerías para mantener el sketch principal limpio. Normalmente se utiliza un archivo .h para las declaraciones y archivos a utilizar (las cabeceras) y los .cpp son las definiciones de esas declaraciones. Para ello ambos archivos deben tener el mismo nombre con la diferencia de la extensión del archivo. Al final, son dos archivos de texto plano que el compilador utiliza para la compilación sin diferenciar su extensión. Nosotros hemos utilizado archivos .hpp, que es una extensión diferente del compilador de C++. En ellos, solo tenemos que definir las funciones y el compilador se encargara de hacer el trabajo sin tener 2 archivos.

Capítulo 7. Proyecto en entorno real

Dado que este proyecto está pensado para un entorno real, vamos a llevarlo a un entorno real con un plano 2D de una vivienda, donde ubicaremos los sensores y actuadores para tener una visión clara de donde estarán.

Detallaremos cada estancia e incluso haremos un bosquejo de posibles mejoras para que queden reflejadas más adelante en su propio apartado del documento. Definiremos las pruebas realizadas y como las hemos llevado a cabo durante la realización del proyecto.

7.1. Descripción.

Disponemos de una casa de 238.6 m2 dispuesta en un salón, una cocina, un baño, un trastero, una habitación, un corredor y un garaje. Se ha decidido que una de las placas de Arduino estará en el trastero junto a la Raspberry PI y otra se ubicará en el salón.

7.2. Plano de la vivienda.

En la siguiente imagen podemos observar la distribución de las funcionalidades (ya que algunos dispositivos como la alarma cuenta con un lcd, un teclado, etc, que no están indicados en este plano, aunque más adelante los enumeraremos todos.

	Sensor de CO2		Lector RFID
	Persiana		Luz
	Calentador		Candado de puerta
	Humedad		Sensor de movimiento
	Conexión WIFI		Plaza de Garaje
	Interruptor		Alarma
	Sensor de Humo		Temperatura

7.3. Funcionalidades

Hemos diseñado estas funcionalidades, siendo a nuestro parecer, las más importantes y que toda casa domótica debería tener. Las funciones como encender la luz con el móvil o controlar la temperatura, todo esto desde cualquier dispositivo, son aspectos de la vida cotidiana que pueden hacer más fácil nuestro día a día, haciéndonos la vida más fácil, cómoda y sencilla.

En nuestro proyecto de casa domótica dispondremos de las siguientes funcionalidades:

- Control por voz: Gracias a Alexa, podremos controlar todos los sensores conectados a openHAB con comandos como: “Alexa, enciende la luz del salón”, o “Alexa, ¿cuál es la temperatura del salón?”. Además, también nos ofrece otras skills para, por ejemplo, tener un hilo musical, leernos las noticias de un periódico, etc.
- Un sistema de seguridad: Dispondremos de un sistema de seguridad mediante una matriz 4x4, una pantalla LCD 16x2 para poder ver los mensajes que nos transmita el sistema, un Buzzer para hacer sonido cuando se active la alarma, dos leds indicando el estado de dicha alarma y un sensor PIR que detectara el movimiento y activara el sistema cuando el estado de la alarma sea activado.
- Control de electrodomésticos: Arduino trabaja a un voltaje y amperaje demasiado bajo para hacer funcionar algo como un calentador eléctrico de agua. Sin embargo, si podemos hacer funcionar electrodomésticos gracias a relés. En este proyecto, para emular dichos electrodomésticos que consumen mucha más energía, hemos utilizado relés. Dichos relés son controlados por openHAB gracias a sus interruptores, pudiendo encenderlos o apagarlos cuando queramos desde cualquier sitio. Es por esto que podríamos instalar un aire acondicionado, y si vemos que hace demasiado calor en la casa antes de llegar, podríamos activarlo y que nos refrescase la casa. En este caso, podríamos hacer alguna rule, ya explicadas anteriormente, para que al alcanzar cierta temperatura se encendiese este relé hasta que nosotros quisiésemos. Esta opción sería muy necesaria en entornos con mucha humedad.

- Control de las luces de la casa: Controlaremos las luces de la casa desde el móvil y cualquier dispositivo con Alexa instalada.
- Sistema de seguridad de Gases: Tendremos un sensor que nos controlara las posibles fugas de gases que tengamos y otro sensor que detectara el dióxido de carbono. Si alguno de los dos es detectado, openHAB nos informara en la aplicación.
- Sistema de control de temperatura y humedad: openHAB nos informara constantemente de la temperatura de los dos sensores de temperatura y humedad ubicados en el salón y la habitación.
- Control de persianas: Nos permitirá controlar si las queremos tener abiertas completamente o cerradas con dos botones, además de si queremos tener un porcentaje abiertas con una barra.
- Control de plaza de garaje: En el garaje tendremos un sensor de ultrasonidos que detectara si hay algún vehículo en la plaza de garaje, y nos indicara si está libre u ocupada.
- Puerta con lector de llaves: La puerta de la entrada solo abrirá con tarjeta autorizadas por nosotros mediante un lector RFID.

7.4. Pruebas realizadas.

Hemos hecho pruebas a cada paso que dábamos para no hacer más trabajo del necesario. Hemos hecho pruebas sobre cada sensor en nuestro Arduino, sobre cada ítem en openHAB, y sobre Mycroft.

Nos hemos encontrado los siguientes problemas:

- En Arduino, nos encontramos que, aunque funcionaban todos los sensores a la perfección individualmente, cuando teníamos todos los sensores funcionando al mismo tiempo, su funcionamiento no era optimo, ya que pulsábamos por ejemplo un sensor táctil para encender las luces y teníamos que estar tocándolo durante al menos 6 segundos para que se encendiese su correspondiente led. Además, cuando pulsábamos una tecla de la matriz 4x4, no era reconocida la pulsación, por lo que no era viable utilizar el sistema de esta manera. Decidimos utilizar dos placas de Arduino MEGA, dividiendo nuestro sketch en dos. Al utilizar librerías para nuestro proyecto, ha sido fácil la transición de una placa a dos, ya que solo hemos tenido que mover los archivos de una carpeta a otra, y incluirlos en el nuevo sketch.
- En Mycroft, después de instalarlo y configurarlo, al no ser compatible nuestro micrófono con Mycroft, después de buscar maneras como buscar los drivers propios y probar más configuraciones, no fuimos capaces de llevar a cabo esta parte del proyecto. Al no querer renunciar de esta parte del proyecto, se buscó otra alternativa, siendo elegida Alexa. Uno de los puntos más importantes de esta elección fue el no tener que estar instalada en la Raspberry PI. Esto implicaba ahorrar tiempo en la instalación y resolución de problemas que podían suponer. Teniendo un dispositivo con Alexa como una Smart TV o un Alexa Dot, ya tendríamos gran parte de los posibles problemas resueltos. Al tener total integración Alexa con openHAB y poder configurarse desde Paper UI, hace mucho más fácil el proceso de configuración.
- En la app móvil de openHAB, hemos probado que enciendan las luces, los relés, nos den la información los sensores de temperatura, gas y de plaza de aparcamiento.
- En cuanto a Alexa, hemos probado los comandos para activar cada sensor y solicitar la información y nos ha respondido con éxito.

Después de resolver dichos problemas y hacer más pruebas de montaje y de programación, podemos ver como no hay más problemas al hacer el montaje final. También hemos hecho pruebas sobre la aplicación de openHAB y hemos observado que todo funciona correctamente comparando los resultados con el terminal serie de Arduino, dado que hemos programado la placa para que nos informe de su funcionamiento desde el terminal.

Capítulo 8. Conclusiones

Nuestro proyecto empezó como una manera fácil y asequible de domotizar cualquier hogar. Fue por estos motivos por los que elegimos tanto hardware como software open-source, siendo un proyecto asequible para cualquier bolsillo, fácil de llevar a cabo por la cantidad de información que podemos encontrar tanto en internet como en documentaciones oficiales. Un usuario medio, debería ser capaz de, siguiendo la documentación, realizar este proyecto.

Además, queríamos acercar el mundo de la domótica a usuarios a los que les podría ser útil pero que no se lo plantean por la posible dificultad que tenga la puesta en marcha del proyecto. Un objetivo es que fuese una interfaz intuitiva para cualquier usuario, que nos dejase ver de un vistazo el estado de cualquier sensor. Para ello, necesitábamos también poder acceder desde cualquier lugar. Hoy en día, más del 68% de la población posee un smartphone. Siendo este hecho una ventaja para nosotros, buscamos poder acceder desde el móvil al sistema. Además de ello, también debíamos poder acceder desde cualquier terminal. openHAB nos ofrecía esta posibilidad, tanto por poder acceder desde cualquier navegador web como por su aplicación nativa.

Otro aspecto que teníamos en cuenta era que fuese posible utilizar cualquier sensor o actuador independientemente de la marca que fuese. Una de las ventajas más grande de openHAB es su sistema de Add-ons. Hay una lista muy grande de sensores compatibles. Además, si alguno no fuese compatible, se pueden buscar maneras de hacerlo compatible, ya que, si son programables, puedes enviar por un protocolo la información por un protocolo que si sea compatible con openHAB y utilizarlo sin problemas.

Este proyecto buscaba facilitar la vida a las personas que quisiesen tener una casa domótica sin tener que hacer reformas costosas, que no fuesen entendidos de la materia pero que por necesidad o por placer, quisiesen entrar en el mundo de la domótica.

Después de finalizar el proyecto, vemos como estos objetivos han sido cubiertos. Hemos podido suplir las necesidades, aun apareciéndonos problemas, y crear un sistema consistente.

La cantidad de información de la que disponen tanto la comunidad de Arduino como la comunidad de openHAB es muy grande y hay verdaderos expertos trabajando para ayudar, resolver dudas y dando ideas creativas para proyectos de gran envergadura. En edades más tempranas de la domótica, era mucho más complicado, ya que era un campo nuevo y no había tanta información como la hay ahora. Si es cierto que había manuales, pero nunca tan completos como lo es internet, y sus respectivas comunidades.

La realización de este trabajo viene dada por la poca información y el gran interés que ha suscitado al autor de este proyecto. El mundo de la domótica era algo que desconocía y ha descubierto aspectos de la informática poco explorados durante el grado. Ha sido interesante programar sobre un dispositivo y gratificante verlo cumplir su propósito y formar parte de un conjunto funcional y con propósito. Hemos descubierto un gran mundo dentro de la informática, con mucho potencial y que realmente quiera continuar trabajando en el proyecto para mejorarlo y ser utilizado en el mundo real.

8.1. Posibles mejoras.

Nuestro proyecto cuenta con las funciones más básicas de una casa domótica, pero es mucho más ampliable. Nos hemos centrado en crear un hogar domótico, dejando la puerta abierta a implementar más funcionalidades, enseñándolas o dando pequeñas pinceladas de como funcionarían.

A lo largo de esta memoria hemos dejado ver algunos aspectos que podríamos implementar a futuro y muchos otros los hemos dejado para este punto. Entre ellos estarían los siguientes:

- Implementación de la puerta del garaje: Como hemos visto, en la puerta de la entrada tenemos un lector RFID. Podríamos implementar otro lector para que este cerca de la puerta de entrada del garaje y otro sensor de ultrasonidos, para que al pasar el lector con una tarjeta valida y estemos lo suficientemente cerca del sensor de ultrasonidos, se abra la puerta del garaje y podamos entrar el vehículo sin necesidad de bajar de él.
- Implementación de un extractor de humos/gas: No es una tarea complicada usando una rule. Tan solo tendríamos que configurar un script que active un relé en caso de fuga de gas o de humo que, al no ser seguro, lo activase y si implementamos un sensor que detecte fuego, activase algún extintor para intentar apagar dicho fuego.
- Alarma con aviso al móvil: Como hemos visto en la rule que hemos implementado, es fácil enviar una notificación push a nuestro terminal. Si conectamos la alarma a openHAB, y la implementamos para que podamos activarla y desactivarla desde el terminal, podríamos hacer también que, si salta la alarma, envíe la notificación a la aplicación y nos avise que ha saltado.
- Usar otro microprocesador: En nuestro proyecto hemos usado dos placas de Arduino MEGA que funcionan a una velocidad de reloj de 16MHz. Aunque es funcional, es mejorable, ya que es un proyecto de una envergadura más grande de lo que una placa de Arduino puede soportar al ser muchas líneas de código y muchas funciones. Es por ello por lo que un procesador como el ESP32, con un microprocesador dual core a 240MHz, podría ser una opción mucho más viable y cómoda. Contaríamos con WIFI encriptado, Bluetooth 4.2, 16 MB de memoria flash, etc.
- Implementación de sensor de luz: Otra posible mejora seria implementar un sensor de luz, que dependiendo de la luz solar que obtenga, suba o baje las persianas, o incluso con una rule, las suba o las baje completamente independientemente de la hora o de la luz.
- Implementar el uso solar para la alimentación de los sensores o de la propia casa: Otro aspecto a tener en cuenta podría ser la alimentación de la casa. Usando una fuente de energía limpia y renovable, el gasto sería bajo o nulo respecto a una casa convencional, pudiendo tener un gráfico de las horas de más uso de electricidad.

Estas son algunas de las posibles mejoras. Hablando de domótica, cualquier sistema es ampliable por la gran cantidad de sensores que existen en el mercado y el abanico de posibilidades es infinito, tanto por funcionalidades como por implementaciones.

8.2. Opinión personal.

Este proyecto ha sido un duro viaje a través del prueba y error. Han sido dos años en los que he ido adquiriendo conocimientos e interesándome por un mundo que a priori no concebía como mío, ya que no me resultaba atractivo desde el principio. Mi prioridad durante el grado era aprender programación, aspectos de ciberseguridad, redes, y nunca me llamo el mundo de la electrónica o el hardware. Después de pasar horas buscando información y de ir probando aspectos pequeños de este mundo, ver cómo funcionan los sensores, trabajar con ellos y programarlos tú mismo, te hace ver que el mundo de la domótica es interesante a la par que útil, hasta el punto de plantearte llevarlo a cabo a gran escala. Algo tan simple como una tira de leds que tenga wifi o bluetooth, la puedes conectar a openHAB y controlarla desde el móvil, desde cualquier lugar, encender el calentador estando en el trabajo, para cuando llegues a casa tener el agua caliente y poder darte una ducha, y un sinfín de posibilidades que pueden hacerte la vida más cómoda.

Durante los cuatro cursos del grado, solo tuve una asignatura de robótica y no pensé que fuese un mundo que me llegase a interesar. Pero después de ver las aplicaciones reales, con las que puedo incluso a nivel personal nutrirme y aplicarlas en mi día a día, empieza a interesarme de verdad este mundo.

En un principio, este proyecto se iba a quedar en un trabajo más de la carrera. No iba a darle vida más allá del trabajo final para cerrar esta etapa. Pero el tiempo invertido en él, me ha hecho darme cuenta de que, con poco dinero, se puede hacer algo realmente útil. Me ha hecho querer ampliarlo para explorar más este mundo para mí, hasta ahora desconocido. Contamos con un sistema funcional, fácil de utilizar y configurable y ampliable con muchas posibilidades.

Habiendo acabado el proyecto, ver el fruto del esfuerzo depositado en él, y ver que realmente funciona, cumple sus propósitos y que el coste de inversión para crearlo es bajo, cumpliendo las expectativas, es muy gratificante y realmente, hace que merezca la pena haber decidido emprender este viaje.

Capítulo 9. Bibliografía y Referencias.

[1] Evolución del porcentaje de hogares con televisores inteligentes en España entre 2015 y 2021. URL:
https://es.statista.com/estadisticas/597365/porcentaje-de-hogares-con-smart-tv-en-espana/

[2]. Azure IoT de Microsoft Cloud. URL:

https://azure.microsoft.com/es-es/overview/iot/

[3]. Apple. Accesorios de domótica. URL:
https://www.apple.com/es/ios/home/accessories/

[4]. Soluciones IoT IBM. URL:
https://www.ibm.com/es-es/cloud/internet-of-things

[5]. Arduino. URL:
https://www.arduino.cc/

[6]. About Arduino. URL:
https://www.arduino.cc/en/about

[7]. ¿QUÉ ES MQTT? SU IMPORTANCIA COMO PROTOCOLO IOT. URL:
https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

[8]. Amazon Alexa Official Site. URL:
https://developer.amazon.com/es-ES/alexa

[9]. Mosquitto Debian repository. URL:
https://mosquitto.org/blog/2013/01/mosquitto-debian-repository/

[10]. Introduction | openHAB. URL:
https://v2.openhab.org/docs/

[11]. Picroft - Mycroft AI. URL:
https://mycroft-ai.gitbook.io/docs/using-mycroft-ai/get-mycroft/picroft#tested-hardware

[12]. Arduino Client for MQTT. URL:
https://pubsubclient.knolleary.net/api#publish

[13]. Memoria EEPROM de Arduino. URL:
https://programarfacil.com/blog/arduino-blog/eeprom-arduino/

[14]. TUTORIAL MÓDULO LECTOR RFID RC522. URL:
https://naylampmechatronics.com/blog/22_tutorial-modulo-lector-rfid-rc522.html

[15]. API Documentation Arduino Client for MQTT. URL:
https://pubsubclient.knolleary.net/api

[16]. Linux - Mycroft AI. URL:
https://mycroft-ai.gitbook.io/docs/using-mycroft-ai/get-mycroft/linux

Vizcaíno, J.R.L. y Sebastiá, J.P. (2014). *Sistemas Integrados con Arduino.* Barcelona, Marcombo.

Imágenes de código y configuración propias.

Capítulo 10. Ficheros de configuración.

Este anexo recogerá el código utilizado tanto en la Raspberry Pi, como en el Arduino. En la Raspberry Pi tendremos un archivo ítems, un archivo sitemaps, un archivo things y un archivo rules.

En cambio, en Arduino tendremos dos archivos .ino y cinco archivos .hpp.

10.1. Raspberry PI

10.1.1. Items

```
  GNU nano 4.8                                                    items/casa.items
Number Temperatura "Temperatura interior [%.1f °C]" <temperature> ["CurrentTemperature"] { channel="mqtt:topic:a31ela68:temperatura" }
Number Humedad "Humedad relativa [%.2f %%]" <humidity> ["CurrentHumidity"] { channel="mqtt:topic:a31ela68:humedad" }
Number TemperaturaHabitacion "Temperatura Habitacion [%.1f °C]" <temperature> ["CurrentTemperature"] { channel="mqtt:topic:a31ela68:temperaturahabitacion"}
Number HumedadHabitacion "Humedad Habitacion [%.2f %%]" <humidity> ["CurrentHumidity"] { channel="mqtt:topic:a31ela68:humedadhabitacion" }
Switch luzsalon "Luz del Salon" <light> ["Lighting"] { channel="mqtt:topic:a31ela68:luzsalon"}
Switch luzcocina "Luz de la Cocina" <light> ["Lighting"] { channel="mqtt:topic:a31ela68:luzcocina"}
Switch luzgaraje "Luz del Garaje" <light> ["Lighting"] { channel="mqtt:topic:a31ela68:luzgaraje"}
Switch luzbano "Luz del baño" <light> ["Lighting"] { channel="mqtt:topic:a31ela68:luzbano"}
Switch luzcorredor "Luz del Corredor" <light> ["Lighting"] { channel="mqtt:topic:a31ela68:luzcorredor"}
Switch luzhabitacion "Luz de la Habitación" <light> ["Lighting"] { channel="mqtt:topic:a31ela68:luzhabitacion"}
String Sensorgas1 "Sensor de Gas 1" <carbondioxide> ["gas"] { channel="mqtt:topic:a31ela68:gas1"}
String Sensorgas2 "Sensor de Gas 2" <smoke> ["gas"] { channel="mqtt:topic:a31ela68:gas2"}
String plazagaraje "Plaza de coche" <"pump"> ["garaje"] { channel="mqtt:topic:a31ela68:plazagaraje"}
Switch calentador "Calentador" <"cistern"> ["Switchable"] { channel="mqtt:topic:a31ela68:calentador"}
Dimmer persiana "Persiana" <"rollershutter"> ["Switchable"] { channel="mqtt:topic:a31ela68:persiana"}
Dimmer persiana2 "Persiana" <"rollershutter"> ["Switchable"] { channel="mqtt:topic:a31ela68:persiana"}
Switch alarma "Apagar Alarma" <switch> ["Switchable"] { channel="mqtt:topic:a31ela68:alarma"}
```

10.1.2. Sitemaps

```
  GNU nano 4.8                                                    sitemaps/casa.sitemap
sitemap demo label="Casa Inteligente" {
    Frame label="Salon" {
        Text item=Temperatura label="Temperatura [%.1f °C]"
        Text item=Humedad
        Switch item=luzsalon icon="light"
    }
    Frame label="Corredor" {
        Switch item=luzcorredor icon="light"
    }
    Frame label="Cocina" {
        Switch item=luzcocina icon="light"
        Text item=Sensorgas1 icon="carbondioxide"
        Text item=Sensorgas2 icon="smoke"
    }
    Frame label="Garaje" {
        Switch item=luzgaraje icon="light"
        Text item=plazagaraje icon="pump"
    }
    Frame label="Baño" {
        Switch item=luzbano icon="light"
        Switch item=calentador icon="cistern"
    }
    Frame label="Habitación" {
        Text item=TemperaturaHabitacion label="Temperatura [%.1f °C]"
        Text item=HumedadHabitacion
        Switch item=luzhabitacion icon="light"
        Switch item=persiana icon="blinds" mappings=[0="Cerrar", 100="Abrir"]
        Slider item=persiana icon="blinds"
    }
    Frame label="Notificaciones" {
        Switch item=alarma
}
}
```

10.1.3. Things

```
 GNU nano 4.8                                    things/mqtt.things
Bridge mqtt:broker:mosquitto "Mosquitto" [ host="192.168.1.146", port=1883, secure=false, username="ubuntu", password="openhabian", clientID="brokerClient" ]
```

10.1.4. Rules

```
  GNU nano 4.8                                    rules/notifications.rules
rule "Send Mobile Push Notification"
when
    Item alarma changed
then
    if(alarma.state == ON)
    {
        logInfo("notifications", "Sending notification via app.")
        sendNotification("vabebal@epsa.upv.es",
            "Se ha disparado la alarma. Verifique y apague la sirena")

        alarma.postUpdate(OFF)
    }
end
```

10.2. Arduino

10.2.1. proto2house.ino

```
1   #include <Wire.h>
2   #include "alarma.hpp"
3   #include "RFID.hpp"
4   #include "persiana.hpp"
5
6   void setup() {
7     Serial.begin(57600);
8     setup_alarma();
9     setup_puerta();
10    setup_persiana();
11  }
12
13  void loop() {
14    loop_persiana();
15    alarmaloop();
16    lecturatarjeta();
17  }
18
```

10.2.2. Persiana.hpp

```
#include <Ethernet.h>
#include <PubSubClient.h>
#include <EEPROM.h>
#define  RELAY1  46     //Conectar al cable de subir persiana
#define  RELAY2  47     //Conectar al cable de bajar persiana
#define segundos 10     // Introduce el tiempo en segundos que tarda la persiana en subir
#define intopic "persiana"

byte mac[]    = {  0xDE, 0xED, 0xBA, 0xFE, 0xFE, 0xED };
IPAddress ip(192, 168, 1, 5);
IPAddress server(192, 168, 1, 146);
//IPAddress ip(192, 168, 0, 5);
//IPAddress server(192, 168, 0, 24);

int paso = segundos * 10;
int retardo = 0;
long lastMsg = 0;
unsigned int posicion = 0;
unsigned int lectura;

EthernetClient ethClient;
PubSubClient client(ethClient);

bool relay_stopped(){
  if (digitalRead (RELAY1) == HIGH && digitalRead (RELAY2) == HIGH){
    return true;
  } else {
    return false;
  }
}

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    String stringOne = "";
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
        stringOne = stringOne + (char)payload[i];
    }
    Serial.println("\n");
  if (relay_stopped()){
        lectura = stringOne.toInt();
        if (lectura >= 0 && lectura < 101) {
            if (lectura > posicion) {
                digitalWrite(RELAY1, LOW);
                retardo = lectura-posicion;
            } else {
                digitalWrite(RELAY2, LOW);
                retardo = posicion-lectura;
            }
```

```
        posicion = lectura;
    }
    EEPROM.put(0, posicion);
  }
}

void reconnect() {
    // Bucle hasta conseguir la reconexión
    while (!client.connected()) {
        // Intento de conexión
        Serial.print("Attempting MQTT connection...");
        if (client.connect("arduinoClient")) {  // Cambiar el nombre si ya hay dispositivos conectados
            Serial.println("connected");
            client.subscribe(intopic);
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000); // Espera 5 segundos para un nuevo intento
        }
    }
}

void setup_persiana(){
  pinMode(RELAY1, OUTPUT);
  pinMode(RELAY2, OUTPUT);
  digitalWrite(RELAY1, HIGH);
  digitalWrite(RELAY2, HIGH);
  client.setServer(server, 1883);
  client.setCallback(callback);
  Ethernet.begin(mac, ip);
  delay(1500);
  EEPROM.get(0, posicion);   // Carga de la EEPROM la posición de la persiana
  if (posicion > 100) {
      posicion = 0;
      EEPROM.put(0, posicion);
    }
}

void loop_persiana(){
  if (!client.connected()) {
        reconnect();
    }
    client.loop();
    if (retardo != 0) {
        delay(paso);
        retardo -= 1;
    } else {
        digitalWrite(RELAY1, HIGH);
        digitalWrite(RELAY2, HIGH);
    }
    // Las siguientes líneas son sólamente para comprobar el funcionamiento
```

```
        long now = millis();
        if (now - lastMsg > 10000) {
            lastMsg = now;
            Serial.print("Posición - ");
            Serial.println(posicion);
        }
    }

```

10.2.3. alarma.hpp

```
#include <Arduino.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <Password.h>
#include <Wire.h>

const uint8_t cols = 4, rows = 4;
const uint8_t PIR = 5, Buzzer = 4, colPins[cols] =  {25, 24, 23, 22}, rowPins[rows] = {29, 28, 27, 26},  LedRojo = 10, LedAzul = 11;
```

```
const char keys[rows][cols] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

uint8_t passPosicion = 0, EstadoBuzzer = LOW, EstadoLed = LOW;
bool EstadoAlarma = false, AlarmaActiva = false, DetectarMovimiento = false;
const uint16_t FuncionDelay = 1000, interval = 400, delayLed = 50;
unsigned long previousMillis;

LiquidCrystal_I2C lcd(0x23, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
Password password = Password("1105");
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, rows, cols );

//Nos mostrar el mensaje estando el sistema activado
void MensajeBloqueo() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Digite su clave");
}

//Esta funcion debloqueara el sistema
void DesbloquearSistema() {
  //The alarm states change to a low state.
  EstadoAlarma = false;
  AlarmaActiva = false;
  DetectarMovimiento = false;
  password.reset();
  passPosicion = 0;
  digitalWrite(LedRojo, LOW);
  digitalWrite(LedAzul, HIGH);
  digitalWrite(Buzzer, LOW);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Sistema desactivado");
  delay(FuncionDelay);
  MensajeBloqueo();
}

//Nos mostrar el mensaje para desactivar la alarma
void MensajeDesbloqueo() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Digite clave para");
  lcd.setCursor(0, 1);
  lcd.print("desactivar alarma");
}

//Esta funcion bloqueara el sistema.
void SistemaBloqueado() {
```

```
62    password.reset();
63    passPosicion = 0;
64    digitalWrite(LedRojo, HIGH);
65    digitalWrite(LedAzul, LOW);
66    //Se activara la alarma dependiendo de i que sera igual a 10.
67    for (uint8_t i = 10; i > 0; i--) {
68      lcd.clear();
69      lcd.setCursor(0, 0);
70      lcd.print("Quedan ");
71      lcd.setCursor(7, 0);
72      lcd.print(i);
73      lcd.setCursor(9, 0);
74      lcd.print(" segundos");
75      lcd.setCursor(0, 1);
76      lcd.print("para la activacion");
77      delay(FuncionDelay);
78    }
79    AlarmaActiva = true;
80    lcd.clear();
81    lcd.setCursor(0, 0);
82    lcd.print("Sistema Activo");
83    delay(FuncionDelay);
84    MensajeDesbloqueo();
85  }
86
87  //Indica que la contraseña no es correcta.
88  void PassIncorrecta() {
89    password.reset();
90    passPosicion = 0;
91    digitalWrite(LedAzul, LOW);
92    lcd.clear();
93    lcd.setCursor(0, 0);
94    lcd.print("Clave incorrecta");
95    for (uint8_t i = 1; i <= 8; i++) {
96      EstadoLed = (EstadoLed) ? LOW : HIGH;
97      digitalWrite(LedRojo, EstadoLed);
98      delay(delayLed);
99    }
100   if (AlarmaActiva == false && EstadoAlarma == false) {
101     digitalWrite(LedRojo, LOW);
102     digitalWrite(LedAzul, HIGH);
103     delay(FuncionDelay);
104     MensajeBloqueo();
105   }
106   else if (AlarmaActiva == true || EstadoAlarma == true) {
107     digitalWrite(LedRojo, HIGH);
108     digitalWrite(LedAzul, LOW);
109     delay(FuncionDelay);
110     MensajeDesbloqueo();
111   }
112 }
```

```
113
114  /*Si la contraseña es correcta y la alarma esta desactivada, llamara a SistemaBloqueado.
115  Si la contraseña es correcta y la alarma esta activada, llamara a DesbloquearSistema.
116  Sino llamara a PassIncorrecta*/
117  void revisarcontra() {
118    if (password.evaluate()) {
119      if (AlarmaActiva == false && EstadoAlarma == false)
120        SistemaBloqueado();
121      else if (AlarmaActiva == true || EstadoAlarma == true)
122        DesbloquearSistema();
123    }
124    else
125      PassIncorrecta();
126  }
127
128  /*keypadEvento tendra en cuenta las pulsaciones introducidas*/
129  void keypadEvento(KeypadEvent key) {
130    switch (keypad.getState()) {
131      /*Si pulsamos mas de 4 teclas llamaremos a revisarcontra*/
132      case PRESSED:
133        if (passPosicion >= 5)
134          revisarcontra();
135        lcd.setCursor((passPosicion++), 2);
136        switch (key) {
137          case '#':
138              revisarcontra();
139              break;
140          case 'A':
141              lcd.clear();
142              lcd.setCursor(0, 0);
143              lcd.print("Digite su clave");
144              password.reset();
145              passPosicion = 0;
146              break;
147          default:
148            password.append(key);
149            lcd.print("*");
150        }
151    }
152  }
153
154  void setup_alarma(){
155    lcd.begin(16,2);
156    //Pin configuration.
157    pinMode(Buzzer, OUTPUT);
158    pinMode(LedRojo, OUTPUT);
159    pinMode(LedAzul, OUTPUT);
160    pinMode(PIR, INPUT);
161    digitalWrite(LedAzul, HIGH);
162    lcd.setCursor(0, 0);
163    lcd.print("Jarvis V 4");
```

```
  lcd.setCursor(0, 1);
  lcd.print(" ");
  delay(3000);
  previousMillis = millis();
  keypad.addEventListener(keypadEvento);
  MensajeBloqueo();
}

//Detectara el movimiento
void MovimientoDetectado() {
  //Activara la alarma y reseteara la contraseña.
  DetectarMovimiento = true;
  EstadoAlarma = true;
  password.reset();
  passPosicion = 0;
  //Indica que se ha disparado la alarma
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Movimiento detectado");
  for (uint8_t i = 1; i <= 8; i++) {
    EstadoLed = (EstadoLed) ? LOW : HIGH;
    digitalWrite(LedRojo, EstadoLed);
    delay(delayLed);
  }
  for (uint8_t i = 1; i <= 8; i++) {
    EstadoLed = (EstadoLed) ? LOW : HIGH;
    digitalWrite(LedAzul, EstadoLed);
    delay(delayLed);
  }
  digitalWrite(LedRojo, HIGH);
  digitalWrite(LedAzul, LOW);
  delay(FuncionDelay);
  MensajeDesbloqueo();
}

/* Esta funcion nos devolvera un estado high o los que se asignara al buzzer*/
uint8_t AlarmaBuzzer() {
  unsigned long currentMillis = millis();
  if ((unsigned long)(currentMillis - previousMillis) >= interval) {
    previousMillis = millis();
    return EstadoBuzzer = (EstadoBuzzer) ? LOW : HIGH;
  }
}

void alarmaloop(){
  /* Si la variable AlarmaActiva esta en un estado HIGH y el sensor PIR detecta movimiento llamara a MovimientoDetectado, que disparara la activacion del buzzer*/
  keypad.getKey();
  if (AlarmaActiva == true && digitalRead(PIR) == HIGH)
    MovimientoDetectado();
  if (DetectarMovimiento)
    digitalWrite(Buzzer, AlarmaBuzzer());}

```

10.2.4. RFID.hpp

```
#include <MFRC522.h>
#include <SPI.h>
#define  RELAY4  7

#define RST_PIN  9    //Pin 9 para el reset del RC522
#define SS_PIN  53   //Pin 10 para el SS (SDA) del RC522 ??????????????
MFRC522 mfrc522(SS_PIN, RST_PIN); ///Creamos el objeto para el RC522

byte ActualUID[4]; //almacenará el código del Tag leído
byte Usuario1[4]= {0xB7, 0x0B, 0xC6, 0x7B} ; //código del usuario 1
byte Usuario2[4]= {0x89, 0x38, 0x3A, 0xB2} ;//código del usuario 2
byte Usuario3[4]= {0x07, 0xD0, 0xF2, 0x20} ;
//////////////////Fin del Control de Acceso//////////////////

 boolean compareArray(byte array1[],byte array2[])
{
  if(array1[0] != array2[0])return(false);
  if(array1[1] != array2[1])return(false);
  if(array1[2] != array2[2])return(false);
  if(array1[3] != array2[3])return(false);
  return(true);
}

void setup_puerta(){
  SPI.begin();        //Iniciamos el Bus SPI
  mfrc522.PCD_Init(); // Iniciamos  el MFRC522
  Serial.println("Lectura del UID");
  pinMode(RELAY4, OUTPUT);
}

void lecturatarjeta(){
  if ( mfrc522.PICC_IsNewCardPresent())
        {
      //Seleccionamos una tarjeta
            if ( mfrc522.PICC_ReadCardSerial())
            {
                  // Enviamos serialemente su UID
                  Serial.print(F("Card UID:"));
                  for (byte i = 0; i < mfrc522.uid.size; i++) {
                          Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
                          Serial.print(mfrc522.uid.uidByte[i], HEX);
                          ActualUID[i]=mfrc522.uid.uidByte[i];
                  }
                  Serial.print("     ");
                  //comparamos los UID para determinar si es uno de nuestros usuarios
                  if(compareArray(ActualUID,Usuario1)){
                    Serial.println("Acceso concedido...");
                    digitalWrite(RELAY4, HIGH);
                    delay(5000);
                    digitalWrite(RELAY4, LOW);
                  }else if(compareArray(ActualUID,Usuario2)){
                    Serial.println("Acceso concedido...");
```

```
                    digitalWrite(RELAY4, HIGH);
                    delay(5000);
                    digitalWrite(RELAY4, LOW);
                  }else{
                    Serial.println("Acceso denegado...");
                  }
                  // Terminamos la lectura de la tarjeta tarjeta  actual
                  mfrc522.PICC_HaltA();
              }
    }
  }

```

10.2.5. tfgprotohouse5.ino

```
#include <Wire.h>
#include "dht11.hpp"
#include "mqtthouse.hpp"

void setup()
{
  Serial.begin(57600);
  setup_ethmqtt();
  setupledbutton();
  setup_calentador();
  setup_dht();
}

void loop(){
  if (!client.connected()) {
    reconnect();
  }
  pulsadortactilpresionado();
  tomardatos(dhtsalon, 0);
  tomardatos(dhthabitacion,1);
  publicarth(tsalon,hsalon,thabitacion,hhabitacion);
  publicargases(MQ2_PIN,"gas1");
  publicargases(MQ7_PIN,"gas2");
  pubgar();
  client.loop();
}

```

10.2.6. mqtthouse.hpp

```
#include <Arduino.h>
#include <Ethernet.h>
#include <PubSubClient.h>
#include <NewPing.h>
#include <EEPROM.h>

#define Trebote 10UL
#define det "Detectado"
#define seg "Seguro"
#define topgas1 "gas1"
#define topgas2 "gas2"
#define ocu "Ocupada"
#define lib "Libre"
#define ts "temperatura"
#define hs "humedad"
#define th "temperaturahabitacion"
#define hh "humedadhabitacion"
#define RELAY3  48
#define TRIGGER_PIN  12
#define ECHO_PIN     11

const int MQ2_PIN = 42;
const int MQ7_PIN = 43;
const int MQ_DELAY = 2000;
byte buttonPin[6] = {34,31,33,30,32,35};
byte ledPin[6] = {36,37,38,39,40,41};
byte lastButtonState[6];
unsigned long timeStamp[6];
const int MaxDistance = 200;
int dist = 0;

byte mac[]    = {  0xDE, 0xED, 0xBA, 0xFE, 0xFE, 0xED };
IPAddress ip(192, 168, 1, 4);
IPAddress server(192, 168, 1, 146);
//IPAddress ip(192, 168, 0, 4);
//IPAddress server(192, 168, 0, 24);

EthernetClient ethClient;
PubSubClient client(ethClient);
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MaxDistance);

void setup_calentador(){
  pinMode(RELAY3, OUTPUT);
  digitalWrite(RELAY3, LOW);
}

void setupledbutton(){
  for (int i = 0; i < 6; i++)
  {
    pinMode(ledPin[i], OUTPUT);
    pinMode(buttonPin[i], INPUT); //INPUT_PULLUP
```

```
52      }
53    }
54
55    void reconnect() {
56      while (!client.connected()) {
57        Serial.print("Intentando conexión MQTT...");
58        if (client.connect("arduinoClient")) {
59          Serial.println("Conectado");
60          client.publish("outTopic","Conectado");
61          client.subscribe("luzsalon");
62          client.subscribe("luzcorredor");
63          client.subscribe("luzcocina");
64          client.subscribe("luzgaraje");
65          client.subscribe("luzbaño");
66          client.subscribe("luzhabitacion");
67          client.subscribe("calentador");
68        } else {
69          Serial.print("Fallo, rc=");
70          Serial.print(client.state());
71          Serial.println(" Intentandolo en 5 segundos");
72          delay(500);
73        }
74      }
75    }
76
77    void callback(char* topic, byte* payload, unsigned int length) {
78      Serial.print("Mensaje Recivido [");
79      Serial.print(topic);
80      Serial.print("] ");
81      String stringOne = "";
82      for (int i=0;i<length;i++) {
83        Serial.print((char)payload[i]);
84        stringOne = stringOne + (char)payload[i];
85      }
86
87      Serial.println("\n");
88      if (strcmp(topic,"luzsalon")==0){
89      if(payload[0] == '1'){
90        digitalWrite(ledPin[0], HIGH);
91        Serial.println(payload[0]);
92        client.publish("outTopic", "Luz Encendida");
93        }
94
95      if (payload[0] == '0'){
96        digitalWrite(ledPin[0], LOW);
97        Serial.println(payload[0]);
98        client.publish("outTopic", "Luz Apagada");
99        }
100     Serial.println();
101     }
102
```

```
103     if (strcmp(topic,"luzcorredor")==0){
104       if(payload[0] == '1'){
105         digitalWrite(ledPin[1], HIGH);
106         Serial.println(payload[0]);
107         client.publish("outTopic", "Luz Encendida");
108         }
109 
110       if (payload[0] == '0'){
111       digitalWrite(ledPin[1], LOW);
112       Serial.println(payload[0]);
113       client.publish("outTopic", "Luz Apagada");
114       }
115     Serial.println();
116     }
117 
118     if (strcmp(topic,"luzcocina")==0){
119       if(payload[0] == '1'){
120         digitalWrite(ledPin[2], HIGH);
121         Serial.println(payload[0]);
122         client.publish("outTopic", "Luz Encendida");
123         }
124 
125     if (payload[0] == '0'){
126       digitalWrite(ledPin[2], LOW);
127       Serial.println(payload[0]);
128       client.publish("outTopic", "Luz Apagada");
129       }
130     Serial.println();
131     }
132 
133     if (strcmp(topic,"luzgaraje")==0){
134       if(payload[0] == '1'){
135         digitalWrite(ledPin[3], HIGH);
136         Serial.println(payload[0]);
137         client.publish("outTopic", "Luz Encendida");
138         }
139 
140       if (payload[0] == '0'){
141         digitalWrite(ledPin[3], LOW);
142         Serial.println(payload[0]);
143         client.publish("outTopic", "Luz Apagada");
144         }
145     Serial.println();
146     }
147 
148     if (strcmp(topic,"luzbaño")==0){
149       if(payload[0] == '1'){
150         digitalWrite(ledPin[4], HIGH);
151         Serial.println(payload[0]);
152         client.publish("outTopic", "Luz Encendida");
153         }
```

```
154
155 ▼   if (payload[0] == '0'){
156       digitalWrite(ledPin[4], LOW);
157       Serial.println(payload[0]);
158       client.publish("outTopic", "Luz Apagada");
159       }
160     Serial.println();
161     }
162
163 ▼   if (strcmp(topic,"luzhabitacion")==0){
164 ▼     if(payload[0] == '1'){
165         digitalWrite(ledPin[5], HIGH);
166         Serial.println(payload[0]);
167         client.publish("outTopic", "Luz Encendida");
168       }
169
170 ▼   if (payload[0] == '0'){
171       digitalWrite(ledPin[5], LOW);
172       Serial.println(payload[0]);
173       client.publish("outTopic", "Luz Apagada");
174       }
175     Serial.println();
176     }
177
178 ▼   if (strcmp(topic,"luzcorredor")==0){
179 ▼     if(payload[0] == '1'){
180         digitalWrite(ledPin[6], HIGH);
181         Serial.println(payload[0]);
182         client.publish("outTopic", "Luz Encendida");
183       }
184
185 ▼   if (payload[0] == '0'){
186       digitalWrite(ledPin[6], LOW);
187       Serial.println(payload[0]);
188       client.publish("outTopic", "Luz Apagada");
189       }
190     Serial.println();
191     }
192
193 ▼   if (strcmp(topic,"calentador")==0){
194 ▼     if(payload[0] == '1'){
195         digitalWrite(RELAY3, HIGH);
196         Serial.println(payload[0]);
197         client.publish("outTopic", "Calentador Encendido");
198         }
199 ▼   if (payload[0] == '0'){
200       digitalWrite(RELAY3, LOW);
201       Serial.println(payload[0]);
202       client.publish("outTopic", "Calentador Apagado");
203       }
204     Serial.println();
```

```
205     }
206   }
207
208   void publicarth(float a, float b, float c, float d){
209     if (client.connect("arduinoClient", "openhabian", "openhabian")) {
210       client.publish(ts,String(a).c_str(),true);
211       client.publish(hs,String(b).c_str(),true);
212       client.publish(th,String(c).c_str(),true);
213       client.publish(hh,String(d).c_str(),true);
214     }
215   }
216
217   void publicarboton (int x){
218     switch(x){
219         case 0:
220         client.publish("luzsalon",String(digitalRead(ledPin[x])).c_str());
221         break;
222         case 1:
223         client.publish("luzcocina",String(digitalRead(ledPin[x])).c_str());
224         break;
225         case 2:
226         client.publish("luzgaraje",String(digitalRead(ledPin[x])).c_str());
227         break;
228         case 3:
229         client.publish("luzbano",String(digitalRead(ledPin[x])).c_str());
230         break;
231         case 4:
232         client.publish("luzcorredor",String(digitalRead(ledPin[x])).c_str());
233         break;
234         case 5:
235         client.publish("luzhabitacion",String(digitalRead(ledPin[x])).c_str());
236         break;
237       }
238   }
239
240   void pulsadortactilpresionado(){
241     for (int i = 0; i < 6; i++)
242     {
243       byte buttonState = digitalRead(buttonPin[i]);
244       if (buttonState == LOW && lastButtonState[i] == HIGH && millis() - timeStamp[i] > Trebote)
245       {
246         digitalWrite(ledPin[i], !digitalRead(ledPin[i]));
247         timeStamp[i] = millis();
248         publicarboton(i);
249         Serial.println("Interruptor Activado");
250       }
251       lastButtonState[i] = buttonState;
252       }
253     }
254
255   void setup_ethmqtt(){
256     client.setServer(server, 1883);
```

```
257     client.setCallback(callback);
258     Ethernet.begin(mac, ip);
259   }
260
261   void pubgar(){
262     dist = sonar.ping_cm();
263     if(dist >= 150 ||  dist == 0){
264         client.publish("plazagaraje",String(lib).c_str(),true);
265         Serial.println(lib);
266       }else{
267         client.publish("plazagaraje",String(ocu).c_str(),true);
268         Serial.println(ocu);
269       }
270   }
271
272   void publicargases(int pin,String topic){
273     bool state= digitalRead(pin);
274       if (!state){
275         client.publish(String(topic).c_str(),String(det).c_str(),true);
276       }
277       else{
278         client.publish(String(topic).c_str(),String(seg).c_str(),true);
279       }
280     }
281
```

10.2.7. dht11.hpp

```cpp
#include <DHT.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#define DHTPIN 2
#define DHTPIN2 3
#define DHTTYPE DHT11

DHT dhtsalon(DHTPIN, DHTTYPE);
DHT dhthabitacion(DHTPIN2, DHTTYPE);
LiquidCrystal_I2C lcdtemp(0x27, 2, 1, 0, 4, 5, 6, 7, 3,POSITIVE);

float tsalon;
float hsalon;
float fsalon;
float thabitacion;
float hhabitacion;
float fhabitacion;

void setup_dht(){
  dhtsalon.begin();
  dhthabitacion.begin();
  lcdtemp.setBacklightPin(3,POSITIVE);  // puerto P3 de PCF8574 como positivo
    lcdtemp.setBacklight(HIGH);   // habilita iluminacion posterior de LCD
    lcdtemp.begin(16, 2);     // 16 columnas por 2 lineas para LCD 1602A
    lcdtemp.clear();      // limpia pantalla
}

void tomardatos(DHT dht, int x){ // si es 0 sera el salon, si es 1 sera la habitacion
  if (x == 0){
  delay(600);

  hsalon = dht.readHumidity();
  tsalon = dht.readTemperature();
  fsalon = dht.readTemperature(true);

  if (isnan(hsalon) || isnan(tsalon) || isnan(fsalon)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  float hif = dht.computeHeatIndex(fsalon, hsalon);
  float hic = dht.computeHeatIndex(tsalon, hsalon, false);

  Serial.print(F("Humedad: "));
  Serial.print(hsalon);
  Serial.print(F("%  Temperatura: "));
  Serial.print(tsalon);
  Serial.print(F("°C "));
  Serial.print(fsalon);
  Serial.print(F("°F  Índice de Calor: "));
  Serial.print(hic);
  Serial.print(F("°C "));
```

```
    Serial.print(hif);
    Serial.println(F("°F"));
    }else if(x == 1){
    delay(600);

    hhabitacion = dht.readHumidity();
    thabitacion = dht.readTemperature();
    fhabitacion = dht.readTemperature(true);

    if (isnan(hhabitacion) || isnan(thabitacion) || isnan(fhabitacion)) {
      Serial.println(F("Failed to read from DHT sensor!"));
      return;
    }

    float hif = dht.computeHeatIndex(fhabitacion, hhabitacion);
    float hic = dht.computeHeatIndex(thabitacion, hhabitacion, false);

    Serial.print(F("Humedad: "));
    Serial.print(hhabitacion);
    Serial.print(F("%  Temperatura: "));
    Serial.print(thabitacion);
    Serial.print(F("°C "));
    Serial.print(fhabitacion);
    Serial.print(F("°F  Índice de Calor: "));
    Serial.print(hic);
    Serial.print(F("°C "));
    Serial.print(hif);
    Serial.println(F("°F"));

    lcdtemp.setCursor(0, 0);
    lcdtemp.print("Grados:  ");
    lcdtemp.print(thabitacion);
    lcdtemp.print(" Cº");
    lcdtemp.setCursor(0, 1);
    lcdtemp.print("Humedad: ");
    lcdtemp.print(hhabitacion);
    lcdtemp.print(" %");
    }
  }

```