DM

Home Assistant versus Hubitat

MASTER DISSERTATION

Josué Duarte de Sousa Ferreira
MASTER IN INFORMATICS ENGINEERING

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

September | 2022

Home Assistant versus Hubitat

MASTER DISSERTATION

Josué Duarte de Sousa Ferreira
MASTER IN INFORMATICS ENGINEERING

ORIENTATION
Karolina Baras

CO-ORIENTATION
Luís Armando de Aguiar Oliveira Gomes

Faculdade de Ciências Exatas e da Engenharia

Tese para a obtenção do Grau de Mestre em Engenharia de Informática

Ano letivo: 2021/2022

Home Assistant versus Hubitat

Josué Duarte de Sousa Ferreira

Orientação:
Karolina Baras

Co-Orientação:
Luís Armando de Aguiar Oliveira Gomes

Setembro 2022

Funchal – Portugal

Resumo

A tecnologia da Internet das Coisas (ou Internet of Things) faz com que cada objeto que esteja à nossa volta permaneça ligado a Internet sem necessidade de qualquer intervenção humana, por outras palavras pode estabelecer a comunicação diretamente com outras máquinas.

A mesma situação acontece nas nossas casas, que começaram a incorporar parte da tecnologia IoT, visto que os dispositivos inteligentes se tornaram cada vez mais acessíveis para qualquer utilizador que esteja interessado. Geralmente, os dispositivos inteligentes fazem parte de apenas uma aplicação de um determinado fabricante, mas opcionalmente, podemos implementar os dispositivos em sistemas operativos que permitem controlar múltiplos dispositivos inteligentes de vários fabricantes.

Este documento explora este tópico dos sistemas operativos para casas inteligentes, estudando os dois sistemas, Home Assistant e Hubitat que foram propostos para o desenvolvimento desta tese. Esta tese fornece uma visão geral do conceito das casas inteligentes e explora os conceitos e os projetos que estejam relacionados com tema, descrevendo as arquiteturas, as vantagens e as desvantagens ao comparar estes dois sistemas.

Palavras-chave: Internet of Things, IoT, Casas Inteligentes, Home Assistant, Hubitat.

Abstract

Internet of Things technology allows each object around of us to remain connected to the Internet without any human interference, that is, objects can now establish direct connections with other “things”.

The same situation happens in our houses as they now incorporate parts of the IoT technology, as intelligent devices become more accessible for any user that is interested. Generally, these devices work only with an application that belongs to a certain manufacturer, but optionally these devices can be implemented in operating systems that are designed to manage multiple devices from different manufactures.

This document explores this topic of operating systems for smart houses, studying two systems: Home Assistant and Hubitat, that were proposed for the development of this thesis. This thesis, gives an overview of the concept of smart house and explores other concepts and projects that are related to these themes, and describes the architectures, the advantages, and disadvantages by comparing of these two systems.

Keywords: Internet of Things, IoT, Smart Home, Home Assistant, Hubitat.

Agradecimentos

Ao longo da escrita desta tese de mestrado, foram vários os esforços que me ajudaram neste percurso, enquanto estudante universitário, mas, também, enquanto futuro Engenheiro de Informática. A minha vida académica, na Universidade da Madeira, foi enriquecida pelas amizades e pelas pessoas que me acompanharam ao longo do curso.

Em primeiro lugar, começo por agradecer aos docentes que me acompanharam no decorrer da realização das Unidades Curriculares, bem como aos meus docentes orientadores, Karolina Baras e Luís Gomes, por toda a dedicação na orientação deste projeto.

Um agradecimento especial à minha família por motivarem o meu percurso académico desde o início da licenciatura e por reconhecerem que persisti intensivamente nos meus estudos académicos. Agradeço à minha namorada, aos meus amigos e aos meus colegas que foram uma força motivadora no decorrer do Mestrado de Engenharia Informática.

Por fim, agradeço, também, aos colegas da empresa ODKAS, por motivarem e estarem disponíveis no desenvolvimento da tese e por fazerem parte do meu desenvolvimento e conhecimento ativo para o meu percurso profissional.

Índice

Índice de Figuras

Lista de Acrónimos

AES – Advanced Encryption Standard

AmI – Ambiente Inteligente

ARM – Advanced RISC Machine

ARPANET – Advanced Reserarch Projects Agency Network

ATM – Automatic Teller Machine

CSA – Connectivity Standards Alliance

CPU – Central Processing Unit

cURL – Client URL

ED – Episode Discovery

ESP – Espressif System

HA – Home Assistant

HAOS – Home Assistant Operating System

HTTP – Hypertext Transfer Protocol

IA – Inteligência Artificial

IEEE – Institute of Electrical and Electronics Engineers

IFTTT – If This Then That

IoT – Internet of Things

JSON – JavaScript Object Notation

LAN – Local Area Network

LED – Light-Emitting Diode

M2M – Machine to Machine

MavHome - Managing An Intelligent Versatile Home

MQTT – Message Queuing Telemetry Transport

OSI – Open System Interconnection

POO – Programação Orientada por Objetos

POS – Point of Sale

QR – Quick Response

REST API – Representational State Transfer Application Programming Interface

RF – Radio Frequency

RFID – Radio Frequency Identification

SDK – Software Development Kit

TCP/IP – Transmission Control Protocol / Internet Protocol

VPN – Virtual Private Network

WAN – Wide Area Network

WPAN – Wireless Personal Area Network

WSL – Windows Subsystem for Linux

1. Introdução

A Internet das Coisas (ou Internet of Things, IoT), descreve-se como uma rede que comunica com objetos físicos. Todos esses objetos estão interligados com sensores, *software* e outras tecnologias que ligam e trocam informações com outros dispositivos e sistemas, através da Internet. A IoT tornou-se numa das tecnologias mais importantes do século XXI, podendo-se aliar com muitas outras tecnologias, tais como, plataformas de computação em nuvem, *machine learning* e a inteligência artificial (IA) conversacional [1, 2]. Por estes diversos motivos, este é um tema recente e que permanece em desenvolvimento.

Como conceito, a IoT não era considerada oficial até 1999. No entanto, a primeira IoT apareceu no início dos anos 80 numa máquina de Coca-Cola, na Universidade de Carnegie Mellon, onde os programadores ligavam a ARPANET (antecessor da Internet) ao aparelho refrigerado, criando uma expressão condicional que verificava se existiam bebidas no frigorífico e se estavam frias, fornecendo a informação aos universitários antes de fazerem o percurso até ao aparelho refrigerado [3, 4].

Oficialmente [5], o conceito foi definido por Kevin Ashton, na altura diretor executivo da Auto-ID, a efetuar uma comparação tecnológica entre os finais do século anterior e o atual. Segundo esse autor, no século XX, todos os dados eram inseridos e modificados por parte dos seres humanos. A partir do século XXI, os dispositivos passaram a colecionar todos os dados possíveis, sem ter alguma interferência por parte dos seres humanos. Por exemplo, um *smartphone* recolhe de forma automática todas as informações que eventualmente iremos precisar: localização, altitude, temperatura, previsão meteorológica, entre outros [3].

Esta tecnologia pode ser aplicada em várias áreas tais como: cidades inteligentes, saúde, vestuário, tecnologias portáteis, transportes, zonas de trabalhos ou mesmo casas, criando as chamadas casas inteligentes [6].

Anteriormente, as casas inteligentes (ou *smart homes*), não eram muito reais, pois eram consideradas essencialmente como conceitos ou imaginações de ficção científica. E refira-se que o conceito não é recente, visto que alguns cientistas desenvolveram ideias que tiveram um papel importante para futuramente criarem modelos de casas inteligentes, tais como: *The House of Tomorrow* (1933), ECHO IV (1969), X10 *Home Automation*, Project (1975), *The TRON* (1989), entre outros [3, 7].

Naturalmente o interesse comercial por esta área foi crescendo, levando a realização de diversos projetos. A título de exemplo, pode-se referir que a própria *Microsoft* tentaram implementar o conceito anteriormente referido. Não era considerado perfeito, contudo,

garantiu o desenvolvimento de certas tecnologias, como por exemplo o *scanner* de código de barras doméstico e o *Time Shift* para a televisão [8].

Atualmente, uma casa inteligente é a integração dos eletrodomésticos, das comunicações e dos equipamentos que são usados para o nosso dia-a-dia num único sistema autónomo. Isto fornece aos donos da casa maior segurança, maior conforto e melhor eficiência energética [9, 10].

Nos últimos anos os avanços nas tecnologias para *smart houses* têm decorrido essencialmente dos desenvolvimentos graças à IoT. Existe também, um grande interesse no mercado, principalmente nas regiões da América do Norte, da Ásia-Pacífico e da Europa Ocidental. No continente asiático, o Japão lidera o mercado graças à sua tecnologia avançada e na Europa Ocidental não está longe devido à sua popularidade no Reino Unido [7, 11].

1.1. Motivação

Com o avanço tecnológico e científico, juntamente com o aparecimento do IPv6 e da IoT, surgem muitas formas para desenvolver uma casa inteligente. E com os protocolos possíveis (por exemplo: Zigbee e Z-Wave), facilita-se a interligação com outros aparelhos inteligentes de uma casa.

No que diz respeito ao *software,* existem dois tipos de aplicações para fazer uma casa inteligente: as aplicações *Single-Tasking* e as aplicações *Multitasking*. As aplicações *Single-Tasking* estão desenhadas para interagir com um dispositivo, enquanto que as aplicações *Multitasking*, estão delineadas para que os utilizadores consigam interagir com múltiplos dispositivos ao mesmo tempo [12].

Desenvolver um sistema operativo para uma casa inteligente torna-se muito dispendioso, e requer uma equipa qualificada e que consiga monitorizar todos os elementos do sistema [11]. No entanto, graças à existência de comunidades de desenvolvimento de *software* e *hardware open-source* e ao processo de desenvolvimento de alguns projetos, especialmente em aplicações para casas inteligentes, acabou por ser possível o desenvolvimento de novos projetos e foi criada uma nova abertura ao mercado das casas inteligentes.

Assim, atualmente, existem muitas aplicações para este fim. As mais populares são: o OpenHab e o Home Assistant. Neste trabalho, inicialmente foram estudadas estas duas opções sendo que por vários motivos, como será descrito na secção 2.6, acabou-se por escolher trabalhar com o Home Assistant.

Posteriormente, e conforme explorávamos a existência de outras aplicações para casas inteligentes (*open-source* ou não), decidiu-se também explorar o Hubitat devido às funcionalidades que fornece aos utilizadores e pela garantia de privacidade e de confiança, uma vez que é um sistema local, ao contrário de outras aplicações para casas inteligentes que recorrem à base na "nuvem".

1.2. Objetivos

Este projeto foi elaborado em conjunto com um colega do curso de Mestrado em Engenharia Eletrotécnica – Telecomunicações da Universidade da Madeira, que se irá concentrar na parte de *hardware*, usando dispositivos inteligentes (sensores, atuadores e entre outros), e construindo um sistema para uma casa inteligente. Por outro lado, este projeto foi desenvolvido no Mestrado de Engenharia de Informática e explora o *software* das casas inteligentes, sendo analisadas as suas funcionalidades e os seus limites.

Este trabalho contém um conjunto de objetivos que foram cumpridos no decorrer da escrita desta tese.

Os objetivos são:

- Compreender o conceito do IoT;
- Compreender o conceito das casas inteligentes;
- Conhecer as soluções existentes que estão no mercado e que recorram ao controlo de dispositivos inteligentes;
- Aprender e compreender os requisitos para uma aplicação de casas inteligentes;
- Integrar na aplicação os dispositivos inteligentes;
- Testar as implementações existentes.

Após a instalação e configuração dos sistemas Home Assistant e Hubitat, pretende-se explorar as funcionalidades, as vantagens e os limites destas aplicações através de um conjunto de cenários possíveis e comparando-as na forma como conseguem interagir.

As questões de investigação são:

- Quais são os limites existentes destas aplicações?
- Que cenários podemos recorrer com estas aplicações?
- Quais são as vantagens da utilização do protocolo MQTT?
- Quais são as vantagens da utilização dos protocolos ZigBee e Z-Wave?

Com estes pontos mencionados, pretendemos adaptar e criar cenários para sistemas sem estar dependente de alguma linguagem de programação para configurar, especialmente para utilizadores finais, pois esses utilizadores habitualmente não têm experiência em programação.

1.3. Contribuição

Para além de contribuir e colaborar com o colega do curso de Engenharia Eletrotécnica – Telecomunicações, através da partilha de documentos de investigação, configurando sistemas de *software* e o *hardware* existentes, conseguimos desenvolver conhecimentos noutras partes, como por exemplo:

- Entender o conceito do IoT;
- Entender o conceito das casas inteligentes;
- Entender o funcionamento dos sistemas Home Assistant e Hubitat;
- Entender funcionamento das automatizações para cada sistema;
- Fazer experiências básicas com Python e YAML, no sistema Home Assistant;
- Estabelecer o conhecimento do MQTT, do ZigBee e do Z-Wave;
- Configurar os dispositivos inteligentes nos dois sistemas;
- Comparar os dois sistemas: Home Assistant e o Hubitat.

Equipamento fornecida pelos docentes da UMa para a realização este projeto:

- Raspberry Pi 3+;
- Hubitat Elevation Home Automation Hub;
- Espressif ESP8266 NodeMCU;
- Sensores AM2302 (DHT22).

1.4. Organização

Este documento está organizado em 5 capítulos: Introdução, Estado da Arte, Desenvolvimento, Testes, Resultados e Conclusão.

O capítulo Introdução, efetua uma breve abordagem do conteúdo sobre o conceito da Internet das Coisas e o conceito das Casas Inteligentes. Para ambos, são resumidamente exploradas as origens, as definições e o que existe atualmente graças ao desenvolvimento tecnológico.

No capítulo Estado da Arte, são analisadas as tecnologias e os protocolos de comunicação existentes para as casas inteligentes, focando especialmente nos protocolos

MQTT e HTTP, juntamente com algumas tecnologias de apoio. Finalmente, são mencionados os trabalhos relacionados, sendo que estes foram separados em dois grupos: as estruturas de casas e as opções tecnológicas. Dentro destes grupos são apresentados artigos, investigações e estudos que motivaram este projeto para casas inteligentes.

No capítulo do Desenvolvimento, são apresentadas as tecnologias que foram usadas com os dois sistemas: Home Assistant e Hubitat. Também são exploradas, resumidamente, as documentações, as arquiteturas e as versões dos sistemas e, ainda, como são configuradas as integrações e as extensões dos mesmos.

No capítulo dos Testes e Resultados mostra-se como foram feitos os testes com os dois sistemas. Sendo assim, são expostos os resultados e é fornecida a comparação final entre os sistemas Home Assistant e Hubitat.

No capítulo da Conclusão é fornecida uma última observação entre os dois sistemas, baseada nas preferências para o utilizador final, e propondo alguns possíveis desenvolvimentos para os dois sistemas, quer na parte de *software* quer na parte de *hardware*.

2. Estado da arte

Este capítulo tem como objetivo mostrar os conceitos da Internet das Coisas e das casas inteligentes e apresentar as tecnologias e protocolos de comunicação possíveis que fortaleçam estes conceitos. Também, iremos explorar os sistemas operativos que facilitam o uso destes conceitos. Para explicar melhor todos estes conteúdos separamos em partes e subpartes.

2.1. Internet das Coisas

A Internet das Coisas ou *Internet of Things* (IoT), segundo a *Oxford Languages* define este conceito como "uma intercomunicação através da Internet de dispositivos de computação introduzidos nos objetos do quotidiano, ativando estes para enviar e receber dados". Igualmente, a Universidade de Cambridge define como "objetos com dispositivos de computação que neles são capazes de interligar entre uns aos outros e trocar dados através da Internet" [18]. Portanto, vai para além das funcionalidades simples da Internet e a considerada "coisa" da Internet das Coisas pode ser uma pessoa com um implante de monitor cardíaco, um animal com um *biochip transponder* ou uma automotiva que transmite dados, que interage com outros objetos e/ou máquinas e contém um conjunto de sensores que notifica o condutor que tem pouca pressão.

Segundo Kevin Ashton, cofundador do *Auto-ID Labs*, quando criou o termo IoT, afirmou que os dados eram quase todos dependentes dos seres humanos, isto é, os humanos criavam dados através de gestos, tais como: escrever, clicar no botão para gravar, aplicar a digitalização de documentos ou fazendo *scan* no código de barras. Todos estes dados equivaliam, aproximadamente, a cinquenta *petabytes*, sendo que um *petabyte* equivale a mil *terabytes* (1 PB = 1000 TB). No entanto, os humanos têm as suas limitações em termos de atenção e precisão. Isto significa que não conseguimos captar todos os dados em tempo real e, se os computadores tivessem todo o conhecimento das coisas, sem haver interferência dos humanos, podiam captar tudo a sua volta, minimizando os custos, as falhas e as perdas [5, 19].

Depois de duas décadas após o aparecimento do termo IoT, a definição do mesmo atualizou e tornou-se num conceito mais concreto. Atualmente, os dados passaram a ser cada vez mais controlados através de sensores e atuadores que estão embebidos nos objetos. Esses mesmos dados são enviados e recebidos entre outros objetos que estão interligados, preferencialmente, por uma rede sem fios de baixo consumo.

As "coisas" começaram a tomar conta do mundo graças à sua distribuição ativa de dados. Isto faz com que as "coisas" se tornam inteligentes, ou seja, a Internet das Coisas não

funciona sem a Inteligência Artificial (IA), que trata de simulações de comportamentos inteligentes e usa os mesmos dados através de técnicas *Machine Learning*, que toma as ações e previsões para melhorar ou otimizar e minimizar os erros. Portanto, o conceito de IoT tem haver com sensores, enquanto o conceito de IA tem haver com comportamentos inteligentes de um ou vários sistemas [18].

2.2. Casas Inteligentes

O conceito de casa inteligente, ao longo destes anos, tem tido um papel muito importante no planeamento e futuro das casas, de maneira a fornecer aos respetivos donos segurança, conforto, eficiência energética e bem-estar [9].

O termo 'casa inteligente' descreve-se como uma residência onde qualquer dispositivo consegue comunicar com outros dispositivos da casa e podem ser controlados remotamente em qualquer ponto da mesma ou até fora de casa com um smartphone. Possivelmente, com este conceito tecnológico pode-se conservar os recursos limitados do planeta Terra.

2.2.1. Origem

As primeiras casas foram consideradas como conceitos, especialmente na ficção científica, exploravam ideias de automação residencial e imaginavam como seria o futuro onde as casas agem de forma autónoma [20].

O ponto da origem deste conceito apareceu graças à revolução industrial e à invenção do controlo remoto feito por Nikola Tesla. No início do século XX, apareceram os primeiros dispositivos para casas, como por exemplo: os aspiradores, as máquinas de lavar roupa, os refrigeradores, entre outros. Podia não ser considerado "inteligente", mas foi um bom começo nesse século e, conforme o tempo passou, a tecnologia tornou-se cada vez mais apta para desenvolver novos conceitos tecnológicos. Porém, ainda estavam muito longe de se tornarem inteligentes [9, 20].

O autor Ray Bradbury do livro *"There Will Come Soft Rains"*, apresentou a ideia de uma casa inteligente que continha funções de assistente de voz para vários fins. Futuramente, esta permanecerá ativa mesmo que os humanos estejam extintos algum tempo, devido a uma guerra nuclear. Embora este conceito forneça conforto e auxílio ativo, também apresenta, ao mesmo tempo, um ambiente inteligente que pode tornar-se solitário e desconfortável [20, 21].

Os conceitos das casas inteligentes que conhecemos atualmente só começaram a surgir nos anos 60 do século XX, sendo o ECHO IV (*Electronic Computing Home Operator*) o

primeiro sistema de automatização. Este computador permitia criar uma lista de compras, controlar a temperatura da casa e também ligar ou desligar os dispositivos [9, 20, 22]. Com o aparecimento desta tecnologia, James Sutherland, um dos engenheiros da ECHO IV afirmou "parece mais uma casa que foi construída para morar um computador em vez do contrário", devido ao seu peso excessivo [22]. Contudo, isto não ia ser a única tecnologia que iria ocupar uma parte ou toda a casa. Nos anos sucessivos, apareceram outros protótipos que melhoraram o conceito de casas inteligentes, como foi mencionado na Introdução.

2.2.2. Atualidade

No início dos anos 2000, rapidamente se tornou popular a tecnologia para as casas inteligentes, devido ao avanço tecnológico e o aparecimento da IoT, o que fez com que suavemente se integrasse nas nossas casas o conceito "inteligente". As casas inteligentes começaram a se tornar como opções acessíveis e, portanto, as tecnologias tornaram-se fiáveis para os consumidores e mais acessível no mercado [9].

A casa inteligente em si cada vez mais se tornou ubíqua, por outras palavras, tornou-se omnipresente enquanto os utilizadores controlam as suas casas juntamente com os seus dispositivos inteligentes e vinte anos depois fez com que existisse a possibilidade de monitorizar os dispositivos mesmo que estejamos distantes das nossas casas [9].

2.3. Comunicação

Com a tecnologia existente temos muitas opções de comunicação quando queremos estabelecer comunicação com os dispositivos IoT. Nesta parte serão exploradas as tecnologias sem fios, como também, protocolos de comunicação: Bluetooth [23,24], Wi-Fi [25, 26, 27], ZigBee [28, 29, 30], Z-Wave [31, 32].

Bluetooth é um protocolo padrão de comunicação projetado para baixo consumo de energia com baixo alcance, que é usado para trocar dados entre dispositivos móveis e dispositivos fixos. O instituto IEEE (*Institute of Electrical and Electronics Engineers*) padronizou o Bluetooth como IEEE 802.15.1, mas atualmente deixou de ser padrão.

O Bluetooth usa ondas de transmissão UHF (*Ultra High Frequency*) entre 2402 MHz e 2480 MHz. Este protocolo pode ser uma boa opção para uma área de rede pessoal (PAN – *Personal Area Network*) e tornou-se também numa das melhores opções tecnológicas para fins de áudio e outras soluções ubíquas, como por exemplo: efetuar chamadas '*hands-free*' nos automóveis. Com o Bluetooth tornou-se numa tecnologia importante para o crescimento da

IoT, pois exige que os dispositivos sejam rápidos e fáceis de comunicar. Com o Bluetooth 5 facilitará a comunicação, duplica a velocidade e aumenta a capacidade de distância com os dispositivos IoT, como também, motiva os utilizadores para este tipo de tecnologia [23,24].

Wi-Fi não é considerado um acrónimo, mas trata-se de uma marca de empresa de marketing. Tecnicamente definimos o Wi-Fi como protocolo IEEE 802.11. Esta ativa as comunicações com os dispositivos *wireless* que estão integrados com o protocolo IEEE 802.11 e incluem os routers e os pontos de acesso. Podemos concluir que este protocolo tem uma grande capacidade para endereçar vários perfis devido a existência de múltiplos padrões desta família de protocolos [25, 27].

Esta família de protocolos torna-se frequentemente comum para rede de área local de dispositivos (LAN – *Local Area Network*), para rede de longa distância (WAN – *Wide Area Network*) e para aceder à Internet, permitindo que haja troca de dados através de ondas de rádio capazes de serem transmitidas entre frequências de 2.4 GHz e de 5 GHz. Estas frequências são mais altas que as frequências que são usadas nos telemóveis [26].

O protocolo anteriormente referido, tornou-se muito comum para muitos dispositivos IoT, particularmente quando estes exigem uma ligação "*machine to machine*" (ou M2M). Contudo, as limitações tornam-se menos prevalentes na escalabilidade e no consumo de energia, como também não se torna numa solução fiável para grandes redes com sensores operados por uma bateria, especialmente num IoT industrial e em cenários de edifícios inteligentes. Outro problema principal de utilizarmos um router que utilizamos no dia-a-dia é a pequena capacidade de distância. Por isso, torna-se mais pertinente para dispositivos que estejam ligados numa tomada elétrica, tais como para aplicações e dispositivos para casas inteligentes [26, 27].

Zigbee trata-se de uma tecnologia sem fios, tornou-se como padrão para IEEE 802.15.4 capaz de operar bandas largas não licenciadas incluindo as frequências 2,4GHz, 900 MHz e 868 MHz. Foi desenvolvido como um padrão global aberto usado para criar uma rede pessoal com rádios digitais de baixa potência especializadas para dispositivos IoT, tais como sistemas de automatização residencial, dispositivos médicos que colecionam dados e outros dispositivos baixa de potência. O protocolo *Zigbee* 3.0 foi criado pelos membros da organização *Connectivity Standards Alliance* [29, 30], acronicamente CSA, (anteriormente chamado como *Zigbee Alliance*) e criaram desde então o padrão existente. Com este novo protocolo foi desenhado uma forma para comunicar dados através de um ambiente '*noisy*' RF que se torna comum nas aplicações comercias e industriais, com esta versão, o que faz com que unifique

todos os dispositivos de diferentes fabricantes numa única rede e torna-se capaz de ser monitorizado por smartphones e tablets numa rede LAN ou WAN [28, 29].

Este protocolo inclui muitas funcionalidades que poderão facilitar a comunicação, tais como: suportar múltiplas topologias, ter uma bateria de baixo consumo que consegue prolongar durante muito tempo, baixa latência, ter encriptação 128-bit AES para estabelecer conexões de dados seguras e alcançar até 65 mil nodos por rede [28].

Z-Wave trata-se de um protocolo de comunicação usado primariamente para casas inteligentes. Este protocolo lidera a tecnologia *wireless* para várias marcas de confiança que estão a trabalhar para que as casas de todos se tornem seguras e inteligentes.

O desenvolvimento deste protocolo foi fundado por seis companhias industriais: Alarm.com, Assa Abloy, LEEDARSON, Qolsys, Ring e Silicon Labs. Com estas empresas criou a aliança *Z-Wave Alliance* com o objetivo de expandir o desenvolvimento tecnológico através deste protocolo. A organização tem um conjunto de objetivos, como por exemplo "Promover a sensibilização do consumidor e o reconhecimento da tecnologia Z-Wave como padrão confiável para o controlo sem fios" [31, 32].

Em termos de frequência acaba por variar conforme os países, por exemplo: E.U.A. e Canada utilizam uma frequência de 908.4 MHz e Israel utiliza uma frequência de 916 MHz [31, 33].

O protocolo Z-Wave recorre ao baixo consumo de bateria e tem outras funcionalidades semelhantes (por exemplo: topologia, consumo e segurança) às do protocolo Zigbee. Portanto, existe uma competição entre protocolos [31, 34].

Em termos de comparação para estes protocolos de comunicação todos estes serão também explorados na secção 2.6.2.2 **Topologias e Protocolos** e comparamos dois outros protocolos HTTP e MQTT na secção 2.6.2.3.

2.3.1. Protocolo HTTP

O protocolo HTTP (*Hypertext Transfer Protocol*), localizado na Camada de Aplicação do modelo OSI (*Open System Interconnection*), trata-se de um protocolo que serve como base que recorre a troca ou informação que vem na *Web*.

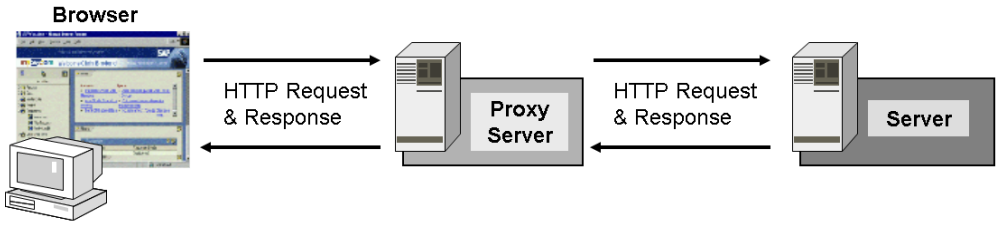

Figura 1. Protocolo HTTP [35]

A arquitetura (Figura 1) recorre a um sistema cliente/servidor que recorre a um método *request-response* para comunicarem entre os mesmos, ou seja, os clientes enviam pedidos HTTP para os servidores e os servidores respondem os mesmos com linhas de estado. As requisições e respostas são enviadas através de agentes usuários ou *proxies*, maioria das vezes este é um navegador da *Web*, contudo, pode ser um robot que preenche as informações necessárias.

2.3.2. Protocolo MQTT

O protocolo MQTT (*Message Queuing Telemetry Transport*) recorre um protocolo de mensagens simples para sensores e pequenos dispositivos moveis que estão otimizados para redes TCP/IP, especializado para a IoT.

Este protocolo utiliza a arquitetura de *publish/subscribe* que transporta as mensagens entre outros dispositivos de forma ordenada e assíncrona. Resumidamente, o sensor ou um dispositivo semelhante, que é um publisher, envia mensagens simples que são destinadas para os clientes, os *subscribers*, esses clientes podem ser computadores, telemóveis ou semelhantes. Entre esta ligação temos um MQTT *broker* (Figura 2), trata-se de um servidor que efetua o envio das mensagens entre os clientes e os sensores.

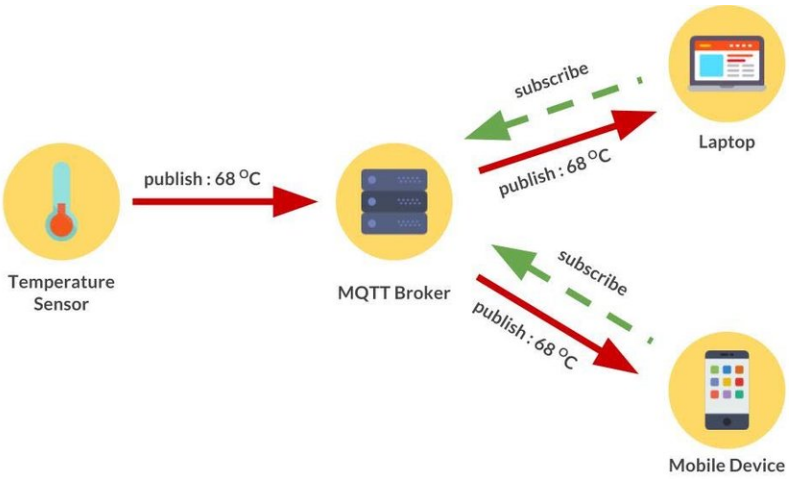

Figura 2. Ilustração da arquitetura pulblish/subscribe [35]

Foi desenhado para estabelecer ligações com localizações remotas onde uma "pequena pegada de código" é recomendada e onde tem uma banda larga limitada. Atualmente é utilizado em várias indústrias, tais como automotivo, manufatura, telecomunicações, entre outros.

2.4. Controlo local vs. controlo via *cloud*

Antes de explorarmos aos sistemas de automação residencial, existe uma distinção importante que devemos saber nos controlos residenciais. As casas inteligentes têm dois tipos de categorias de Hubs: controlo à base de *cloud* e controlo local [36].

Para diferenciarmos entre o controlo local e o controlo à base de *cloud*, devemos imaginar quando queremos ligar as luzes de uma casa. Para ativarmos as luzes, não exige ter ligação a internet, basta só mandar uma mensagem para ativar os dispositivos. Caso fosse fora de casa, o utilizador pode ativar remotamente as luzes antes de chegar a casa [37].

Podemos observar que ambos os sistemas têm as suas vantagens [38]. O controlo local não exige nenhuma partilha de informação ou fuga de informação, tem uma resposta rápida e também é confiável. No entanto, o controlo tem duas limitações: não funciona remotamente e tem poucas funcionalidades. Para o controlo via *cloud*, temos possibilidade de comunicar remotamente com os dispositivos e a *firmware* sempre será atualizar para prevenir falhas de segurança e para adicionar novas funcionalidades que podem ser assistentes de voz, rotinas, IFTTT e entre outras funcionalidades.

Cada vez mais as casas inteligentes utilizam controlos via *cloud* devido a dois fatores [38]: tem a sua facilidade de conectar entre os utilizadores e com os dispositivos inteligentes na maioria das vezes esses dispositivos são universalmente compatíveis e não exige programar cada dispositivo e a *cloud* consegue lidar com processamento mais complexo, enquanto analisa os dados.

Apesar de existir constante debate entre estes dois controlos, em termos de preferência para uma casa inteligente, no entanto, ambos são ideais conforme os utilizadores desejam para as suas casas, provavelmente seria perfeito ter um sistema híbrido que contenha ambos os controlos [38].

2.5. Sistemas de automação residencial

Um sistema de automação residencial, também conhecido como domótica, é uma solução tecnológica que facilita a automação para eletrodomésticos, aplicativos e outros

dispositivos eletrónicos dentro de casa. Este sistema utiliza uma combinação de tecnologias *hardware* e *software* que permitem o controlo e a gestão através das aplicações e os dispositivos que estão em casa [39, 40].

Tipicamente um sistema de automação procura estabelecer a ligação de todos os dispositivos através de um *hub*, que por vezes são chamados como *gateway*. A interface do utilizador tem formas para fornecer o controlo da casa. Podemos recorrer estas interfaces em terminais, em tablets, em computadores com uma interface *Web* ou com uma aplicação de um telemóvel. Muitas vezes, a gestão e o controlo dos dispositivos e dos eletrodomésticos têm um conjunto de ações pré-definidas, através de um conjunto de comportamentos definidos pelo sistema ou pelo utilizador. Possivelmente podem ser controlados remotamente mesmo fora de casa [39, 40, 41].

Para desenvolver uma casa inteligente temos muitas opções e acaba por ser caro, contudo, ao construirmos estes sistemas contribui-se economicamente para que o utilizador recorra a estes eletrodomésticos graças às suas funcionalidades inteligentes, quer no uso da eletricidade, quer no uso de água potável. Podem, ainda, contribuir ecologicamente para a preservação dos recursos naturais do planeta Terra. As outras contribuições que podem fornecer ao utilizador são o conforto e a segurança, no entanto, a segurança tem que ser atualizada, devido aos *leaks* de segurança entre o sistema e o utilizador [40, 41, 42].

2.5.1. Home Assistant

O sistema Home Assistant (HA) é um dos sistemas de software open-source mais populares. Tem como funcionalidade de um *Central Hub* que controla e monitoriza os dispositivos compatíveis e foca no controlo local e na privacidade. Podemos aceder este sistema através de um browser ou usando aplicações de Android e iOS. Também podemos usar comandos através dos assistentes virtuais como Google Assistant e Amazon Alexa [43, 44].

Uma vez que o sistema é *open-source*, existem várias formas para este ser instalado. Atualmente, podemos instalar em qualquer sistema operativo. Os desenvolvedores criaram um próprio sistema operativo – Home Assistant Operating System (HAOS) – e recomendam que seja instalado para computadores de tamanho reduzido tais como Raspberry Pi, Odroid e ASUS Tinkerboard [45].

2.5.2. OpenHab

O sistema OpenHab é um sistema *open-source* popular, considerada também um dos sistemas rivais para o Home Assistant. O sistema consegue integrar em vários sistemas e tecnologias, afirmando que suporta milhares de dispositivos. O servidor pode ser implementado em qualquer sistema: Linux, macOS, Windows, Raspberry Pi, Docker, etc. Para facilitar o uso deste sistema, podemos implementar o sistema 'openHABian' num Raspberry Pi ou noutro dispositivo semelhante usando um cartão SD, tal como acontece com Home Assistant OS [46].

Dentro do sistema podemos implementar um sistema completamente privado, não exigindo o uso de serviço *cloud* para funcionar, contudo, podemos adicionar opcionalmente um sistema '*Cloud-Friendly*' tais como Google Assistant, Amazon Alexa, Apple HomeKit e entre outros sem nenhuma restrição. Por este motivo, tendemos a comparar as funcionalidades entre OpenHab e o sistema rival - Home Assistant, que ocorreu numa investigação [47] e também adicionamos outras investigações realizadas [48, 49, 50] na Universidade da Madeira, na Unidade Curricular Computação Ubíqua. Todos estes são mencionados nos Trabalhos Relacionados (2.6.2 **Opções Tecnológicas**).

2.5.3. Hubitat

Trata-se de um sistema de automação criada pela companhia Hubitat Elevation que tem intenções de fornecer a maior privacidade possível aos utilizadores, pois não exige nenhum serviço via *cloud*. Para obter este sistema temos que comprar o produto, logo não exige nenhuma instalação ou configuração por parte do utilizador [51].

Em contraste, enquanto defendem o conceito de 'completa privacidade', conforme analisamos no capítulo seguinte, para termos completo acesso ao sistema do Hubitat e para começar a utilizar os dispositivos para o mesmo, têm que criar uma conta.

2.5.4. SmartThings

A aplicação SmartThings [52, 53] foi criada pela Samsung e garante o controlo de todos os dipositivos que foram instalados numa casa. Este tipo de aplicação não está exclusivamente limitado para dispositivos da Samsung.

Escolhemos esta aplicação porque tem algumas comparações que são capazes de diferenciar com o sistema Hubitat [54, 55].

2.6. Trabalhos Relacionados

Nesta subsecção juntamos artigos e trabalhos de investigação que estão relacionados e que interligam o projeto que estivemos a desenvolver. Dividimos em duas partes. A primeira parte – 'Estrutura de casas inteligentes' e a segunda parte – 'Opções tecnológicas'.

A primeira parte explora o conceito das casas inteligentes, onde existe um conjunto de dispositivos que estão aptos para cenários apropriados. A segunda parte explora opções tecnológicas que podemos recorrer e comparar através do nosso projeto.

2.6.1. Estruturas de casas inteligentes

Quando pensamos em estruturas de uma casa inteligente, pensamos numa construção de raiz, para caber o máximo de dispositivos possíveis dentro de casa, recorrendo a uma arquitetura exclusiva ou diferente do costume. Esta subsecção anota certas situações, quando estamos a aplicar corretamente ou não, e enquanto estamos a desenvolver uma casa inteligente.

2.6.1.1. *Gator Tech Smart House*

O artigo "*Gator Tech Smart House*" [56], no tempo que foi publicado, descreve que, anteriormente, os projetos centravam-se em sistemas básicos, interligando com sensores, atuadores e outros dispositivos no ambiente. Contudo, existem muitos sistemas de computação que não têm capacidade de evoluir à medida que as novas tecnologias se emergem, integrando vários elementos em que os processos são, maioritariamente, feitos manualmente. Quando inserimos um elemento, devemos investigar as suas características, como podemos configurar e integrar. Provavelmente pode causar conflitos ou comportamentos anormais dentro do sistema.

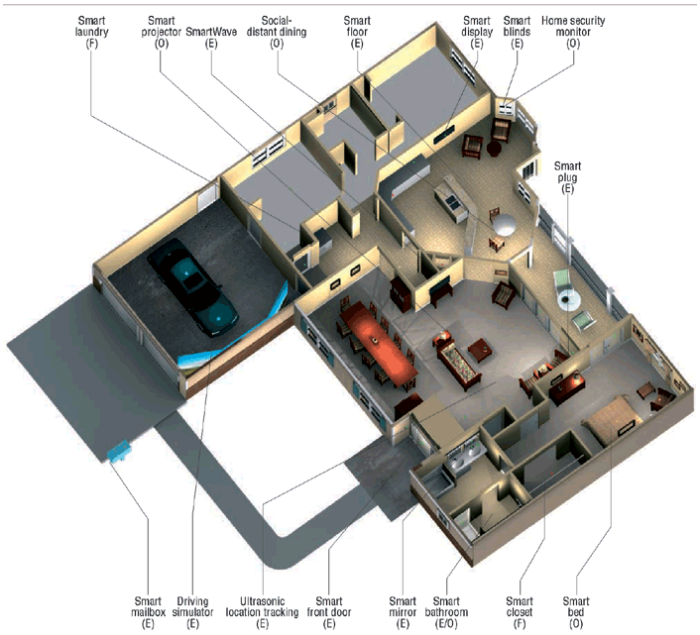

Figura 3. Gator Tech Smart House [56]

Para resolver essa situação, o Laboratório de Computação Móvel e Persuasiva da Universidade da Flórida desenvolveu um espaço programável, nas quais trata-se de um Ambiente Inteligente (ou AmI), que existe num ambiente executável e na biblioteca de software. Os protocolos de *gateway* integram automaticamente os componentes enquanto recorrem, através de um *middleware* genérico que mantém a definição do serviço para cada sensor e atuador no espaço. Os programadores montaram serviços em aplicativos compostos em serviços de terceiros podem ser implementados ou estendidos facilmente [56].

Acabou por ser desenvolvido a casa inteligente *Gator Tech Smart House*, que contém vários sensores em vários locais como a cama, o chão, a caixa de correio, as portas, entre outros. Esta tecnologia facilita os idosos e as pessoas portadoras de deficiência quando estão sozinhos nas casas inteligentes. Contém sensores ultrassónicos que detetam todos os movimentos para determinar a localização dos mesmos. Múltiplos recetores são ativados e detetam a variedade de distâncias para serem calculados por cada recetor e ter a melhor localização possível para os utilizadores [56, 57].

Com o passar do tempo há cada vez mais estes conceitos e atualmente acabou por se tornar mais acessível no mercado. Existe uma grande variedade de eletrodomésticos e *framework* de *software*, de muitas marcas conhecidas:

- Philips Hue [58], fornece lâmpadas inteligentes aptos para contribuir os utilizadores através de cenários personalizados. Também existem sensores de movimento e sensores externos para minimizar a interação dos utilizadores, garantido também eficiência energética e segurança para os donos da casa. Existem, ainda, outras funcionalidades

que aumenta o interesse para o utilizador enquanto mistura com os produtos de outros fabricantes diferentes.

- Samsung SmartThings [53], trata-se de uma aplicação que consegue gerir uma casa com os múltiplos dispositivos da Samsung SmartThings e de outros fabricantes. Opcionalmente, podemos adicionar mais do que uma casa facilitando assim controlo das casas numa só aplicação. Também tem funcionalidades para automatizar condições enquanto os utilizadores não estiverem presentes.
- Apple HomeKit [59], uma *framework* de *software* exclusivamente para os dispositivos da Apple (iOS, iPadOS, macOS, watchOS e tvOS) que permite os utilizadores controlarem e configurarem com outros eletrodomésticos inteligentes de múltiplos fabricantes com a ajuda da assistente de voz Siri.

Também existem outros exemplos que utilizam funcionalidades semelhantes, tais como por exemplo: Google Assistant, Xiaomi Mi Smart Home, mydlink, entre outros.

A arquitetura *middleware* desta casa inteligente contém várias camadas: física, plataforma de sensor, aplicação, serviço, conhecimento e gestão de contextos. Na camada física contém elementos físicos: eletrodomésticos (tais como: ar condicionado e termostato), dispositivos inteligentes (tais como: sensores e atuadores) e o sistema de segurança. Na camada da plataforma do sensor, nem todos os objetos estão interligados e cada um define uma relação de espaço persuasivo, nesta plataforma pode comunicar com os dispositivos existentes e converte qualquer sensor e atuador da camada física como um serviço de *software*, isto não exige os *developers* a tomarem conhecimento da camada física. A gestão de contexto permite os programadores a criar e registar contextos.

Para além de tomarmos conhecimento da existência destas camadas, percebemos que para ser programado e para que funcione o sistema do Gator Tech exige três atividades inteligentes:

- Engenharia de contexto, interpreta os dados e identifica os estados de alto nível, como por exemplo, identificar o “quente” e o “frio”;
- Engenharia de *software*, descreve os vários comportamentos dos componentes de *software*, como por exemplo ligar o aquecedor ou criar um conjunto de ingredientes para o menu;
- Associando os comportamentos com o contexto, esta define quais são as peças de *software* que podem executar num contexto particular e quais são as peças do sistema que devemos invocar sobre uma mudança contextual.

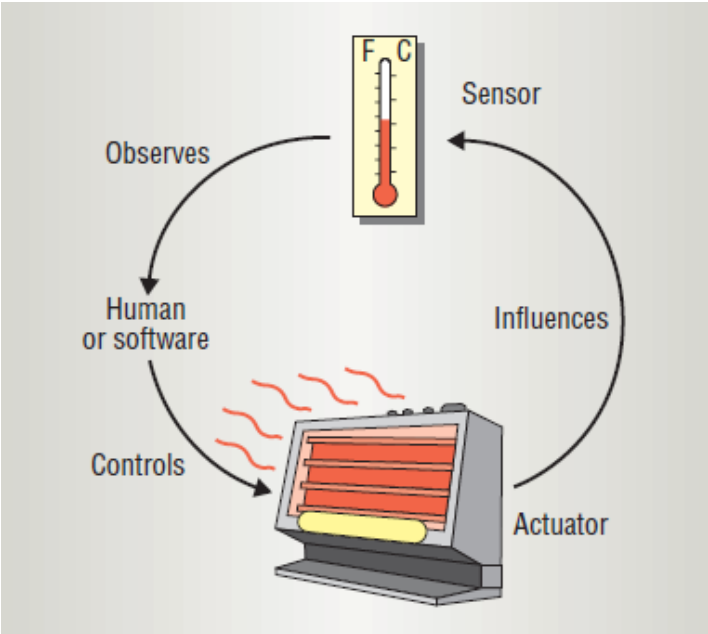

Figura 4. Processo de observação/controlo entre atuador e sensor [56]

Como podemos ver na Figura 4, representa o processo de observação/controlo, onde mostra a interação entre os sensores e os atuadores. As casas inteligentes devem obter informações através dos dispositivos inteligentes e efetivamente devem tomar ações, como por exemplo, nas condições da temperatura do ambiente, se estiver frio, o sistema deve ativar o termostato para manter uma temperatura confortável.

Os investigadores programaram uma plataforma que integra todos os eletrodomésticos, sensores e atuadores e outros dispositivos neste ciclo de observação e controlo. Por exemplo, ao instalar a plataforma do sensor, cada sala fica interligada com os sensores de humidade e de temperatura e o programa descodifica as informações dos mesmos. Cada sala é representada como um nó de uma árvore de um sistema que controla a distribuição da temperatura e da humidade de cada um destes.

2.6.1.2. *The TRON*

O projeto THRON [60] baseava-se na criação de um conceito de uma "arquitetura total de computador" para que sirva como base de construção para uma sociedade que se interliga com a tecnologia computacional do século XXI. O desenvolvimento e a investigação estavam divididos em duas partes: projetos de tecnologia básica e projetos de aplicação tecnologia.

Segundo o conceito chave do projeto, recorre-se a uma fusão entre humanos, natureza e computadores, e assim, frui uma dessas ideias que foi '*The TRON Intelligent House*'. Mais

de 15 companhias japonesas participaram no projeto e dentro desta casa continha um total de 380 computadores.

A casa estava completamente preenchida por computadores que continham informações externas (televisão, rádio, telefone) e internas (sistema audiovisual, *intercom*, sensores de segurança), que eram canalizados em unidades de exibição para cada sala. Também havia dezenas de altifalantes, um controlo remoto de uso geral e um interruptor de painel padronizado.

Mesmo que esta casa tivesse muitos métodos inovadores e mesmo que o projeto fosse muito ambicioso, aguentou apenas três anos para aplicar experiências que era organizado por vários grupos tecnológicos. Também acabou por ser uma casa muito dispendiosa, uma casa à frente do seu tempo, e por ser grande polémica à imprensa japonesa [61]. Portanto, este projeto mostrou uma forma de como não desenvolver uma casa inteligente.

2.6.1.3. MavHome

O projeto MavHome (*Managing An Intelligent Versatile Home*) [62], criado na Universidade do Texas teve como objetivo de desenvolver uma casa que comporta como um "agente racional", tentando perceber e adivinhar os comportamentos ou padrões que vem dos habitantes da casa através dos sensores e atuando no ambiente através desses dispositivos.

No documento da investigação, os investigadores narram o funcionamento do MavHome enquanto interagem com o utilizador "Bob" com as seguintes funcionalidades internas e externas da casa: o termostato, as luzes, o alarme, a máquina de café, o irrigador automático e a previsão meteorológica [62].

A arquitetura do MavHome exige alguns requisitos para obter uma funcionalidade semelhante ao que foi narrado. Nesta arquitetura deve ser integrada tecnologias recorrendo à base de dados, à robótica, à máquina de aprendizagem, à computação móvel e à computação multimédia. Caso haja ambientes alargados o agente do MavHome divide as suas tarefas hierarquicamente. A arquitetura do agente (Figura 5) tem um conjunto de cada:

- Camada de decisão, que seleciona as ações para que o agente as execute baseado nas informações fornecidas das outras camadas, através da camada de informação;
- Camada de informação que obtém, armazena e gera conhecimento útil para tomar decisões;
- Camada de comunicação que facilita a comunicação de informação, pedidos e questões entre os agentes;

- Camada física que contém o *hardware* que está dentro de casa incluindo dispositivos individuais, transdutores e *hardware* de rede.

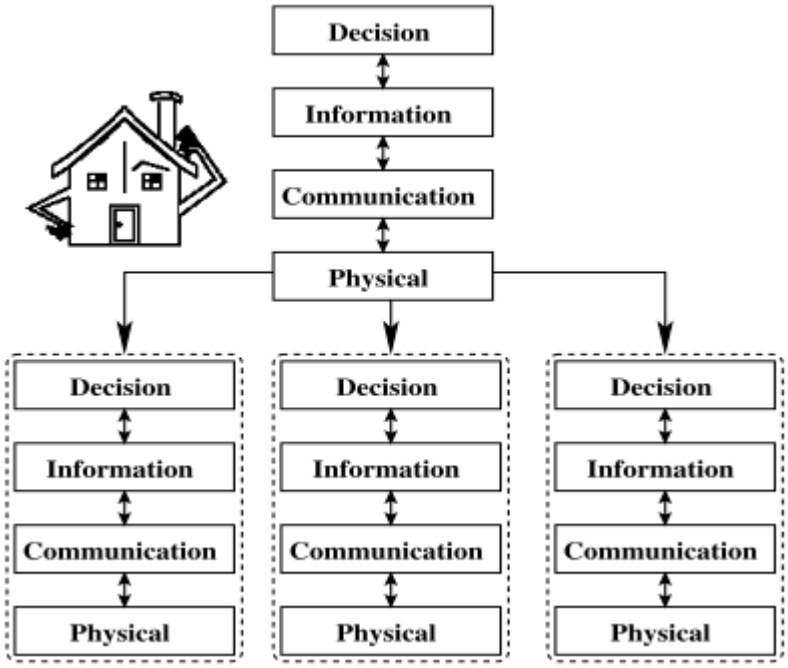

Figura 5. Arquitetura do agente MavHome [62]

Em termos de funcionalidade dos agentes MavHome a perceção é feita de forma ascendente. Os sensores transferem as informações para o agente e, se for necessário, envia os mesmos dados para outros agentes, através da camada de comunicação. A base de dados grava os dados na camada de informação, atualiza os conceitos de previsão e decisão e, consequentemente, alerta a camada de decisão a existência de novos dados.

Durante a ação que fora executada, a camada de decisão seleciona a ação e relaciona as decisões na camada da informação. Depois da base de dados ser atualizada, a camada da comunicação direciona as ações para indicar a célula mais apropriada para executar. Se a célula for outro agente, o agente que recebe a informação da mesma célula, deve decidir a próxima ação. Uma interface especializada fornece capacidades de interação com os utilizadores e com recursos externos, como por exemplo a Internet. Os agentes comunicam com outros agentes através de um fluxo hierárquico de controlo e informação [62, 63], como podemos ver na Figura 5.

Com este desenho da arquitetura, tende a aumentar o conforto dos seus utilizadores enquanto reduz o custo das operações, um agente racional tem de prever os padrões de mobilidade e o uso dos dispositivos dos habitantes [56, 62].

Para que haja funcionamento como um ambiente inteligente, deve ser adquirido e aplicado o conhecimento de forma que se adapte aos habitantes para alcançar esses objetivos

de conforto e eficiência. Com essas capacidades, obtidas através de algoritmos de previsão eficazes, e com as atividades dos habitantes, o MavHome pode decidir se deseja ou não automatizar a atividade ou encontrar outra maneira para melhorar a atividade para alcançar os objetivos na casa [62].

Na experiência deste projeto são recorridos três algoritmos:

- Algoritmo SHIP, fornece uma combinação adequada dos eventos mais recentes que foram obtidos anteriormente;
- Algoritmo Active LeZi (ALZ), utiliza os princípios da teoria da informação para processar sequencias de ações anteriores, recorrendo à interação habitante-dispositivo como uma cadeia de eventos de Markov;
- Algoritmo TMM identifica tarefas de alto nível numa ação de sequências para ajudar a criação de um Modelo de Markov que aplica a previsão de ações.

Todos estes algoritmos são considerados uteis para identificar as atividades dos habitantes numa casa inteligente. As informações podem ser utilizadas para automatizar as interações dos dispositivos, assim, remove necessidade de interagir manualmente os dispositivos. Caso aconteça alguma previsão errada, para além de ser inconveniente para o morador, este tem de reverter a ação para que a casa execute corretamente, caso contrário, tem de reparar as funcionalidades da casa devido a esta falha.

Para prevenir este problema de identificar e automatizar para cada habitante, os investigadores recorreram um algoritmo alternativo que recorre à *data mining* chamado ED (*Episode Discovery*). Este algoritmo identifica "episódios" significantes através de um "histórico" por parte de um dos habitantes. Um "episódio" pode ser visto como um conjunto relacionado de eventos do dispositivo, este pode ser ordenado, parcialmente ordenado e não ordenado. Este "episódio" pode ocorrer através de um intervalo regular ou resposta de outros "episódios" chamamos estes como trincos ou *triggers*. As ações são logo automatizadas e basedas numa significância de padrões minerados, tal como uma previsão precisa para o próximo evento.

Para melhorar a qualidade das previsões, as sequências das ações devem ser filtradas usando o algoritmo ED e se a sequência for significante, então as previsões podem ser criadas conforme são aplicadas as ações. Portanto, usando este algoritmo implica o aumento da previsão conforme são usados os dispositivos sequencialmente.

Após a analise de todos os algoritmos existentes, especialmente o algoritmo alternativo ED, os investigadores conseguem provar que existem muitas abordagens para aplicar previsão

numa casa inteligente. Também apontaram pontos fortes das abordagens alternativas, considerando o algoritmo principal do *MavHome* como um algoritmo '*meta-preditor*', chamado *Predict*.

O algoritmo *Predict* usa uma rede neuronal de retro propagação capaz de aprender e obter o valor de confiança para cada previsão algorítmica fundamentado nos dados recolhidos e na precisão do algoritmo, também são acompanhados com meta recursos tais como os dados experimentais (*training data*), número de dispositivos no ambiente, número de habitantes e a significância de eventos determinados pelo algoritmo ED.

A investigação apresenta uma arquitetura que permite que a casa se torne num agente inteligente e muitos dos algoritmos são implementados para executar papeis importantes num ambiente adaptativo e automatizado. Os resultados indicam que as precisões da previsão são altas, mesmo que haja outras atividades e com o algoritmo *Predict* permite que cada uma das abordagens de previsão desempenhem um papel na previsão da atividade dos habitantes.

2.6.2. Opções Tecnológicas

Atualmente, com a tecnologia existente temos a possibilidade de desenvolver várias formas para construir e testar juntamente com os dispositivos inteligentes de diversos fabricantes numa casa inteligente. Esta parte apresenta um conjunto de investigações e de conceitos, as tecnologias que nos representam e como conseguimos controlar os dados dos dispositivos existentes.

2.6.2.1. Ambiente Inteligente

Com o aparecimento do termo Computação Ubíqua, criado por Mark Weiser [64], forneceu uma nova visão para a computação móvel para os dispositivos móveis e os serviços tornam-se ubiquamente disponíveis, fornecendo o acesso às redes de dados de forma segura [65].

Segundo Weiser afirmou que "As tecnologias mais profundas são aquelas que irão desaparecer. Essas se entrelaçam no fabrico quotidiano até que sejam indistinguíveis disso" [64]. Corretamente tinha previsto os telefones à base de satélite e a Internet são exemplos de tecnologias profundas e invisíveis, logo, no próximo futuro o Ambiente Inteligente será igualmente profundo e invisível.

A visão do Ambiente Inteligente foi uma proposta para as ideias iniciais da Computação Ubíqua. Weiser antecipou um mundo digital onde os dispositivos eletrónicos estão integrados numa rede distribuída [64, 65].

O AmI (Ambiente Inteligente) refere-se a ambientes eletrónicos que são sensíveis, inteligentes e responsivos na presença das pessoas. A ocorrência da visão AmI apareceu pelo facto que o desenvolvimento tecnológico atual fez com que a eletrónica se integra ao meio ambiente, por exemplo a interação das pessoas com os objetos no seu ambiente de forma confiável e mais natural possível. O AmI deve interagir, adaptar, precipitar e estar pronto para responder às necessidades dos utilizadores [64].

2.6.2.1.1. Definição

A palavra 'ambiente' refere-se à necessidade de incorporar a tecnologia em grande escala de forma que se integra de maneira não intrusiva no nosso dia-a-dia. A palavra 'inteligente' refere-se ambientes digitais que exibem formas específicas de interação social, como por exemplo os ambientes devem ser capazes de reconhecer as pessoas, personalizarem-se os gostos dos utilizadores e adaptarem-se conforme passa o tempo [64].

2.6.2.1.2. Próximo passo para Inteligência Artificial

Para alguns sistemas expõem AmI como uma '*buzzword'*, uma palavra da moda, integrando um pequeno pedaço da inteligência, certos investigadores implementam um sistema AmI sem IA (Inteligência Artificial), centrando apenas nos sensores, atuadores, comunicações e computação ubíqua. Os investigadores [66] apontam uma visão diferente, indicando que existe uma grande importância no uso da IA num sistema AmI. O ambiente pode variar, pois pode ser numa casa, num carro, num museu, num escritório, etc. Quando implementamos um AmI, este deve receber informações dos utilizadores e interagir com os mesmos, através dos vários sensores ultrassónicos, microfones e câmaras.

De certa forma, percebemos que o AmI se torna importante no novo desafio e faz parte da evolução para IA [67], pois com os métodos e as técnicas da IA, conseguem cumprir tarefas importantes para o ambiente. Tarefas que se envolvem com vários *inputs* que os humanos recorrem. As primeiras tarefas que a IA deve reconhecer são os meios de comunicação que os humanos interagem frequentemente: a escrita e a fala. A outra tarefa essencial que AmI deve perceber é a visão humana, pois torna-se num dos sensores mais ricos dos seres humanos, para os computadores compromete em muitas áreas: aquisição de imagem, processamento de

imagem, reconhecimento de objetos (2D ou 3D), análise de cenário e análise de fluxo de imagem. Conforme passa o seu tempo de desenvolvimento, os investigadores e os utilizadores esperam que o AmI consiga receber e perceber todos os meios de comunicação [66].

Portanto, os sistemas AmI devem ser capazes de interagir inteligentemente com os humanos, pois qualquer interação exige consistência e contexto. O contexto envolve-se em fatores, como interfaces de iniciativa mista, que se adaptam aos utilizadores em determinadas situações. Também deve aprender enquanto interage com o utilizador, tomando consciência durante a situação com o mesmo, escalando assim a sua inteligência. Como estes sistemas interagem com os utilizadores, precisam de considerar a existência de fatores sociais e emocionais, percebendo também o interesse pessoal, o interesse social e o estado emocional dos seres humanos [66].

2.6.2.1.3. Sistema versus Inteligência Social

Na visão do AmI [65], esta precisa de elementos-chave dos seres humanos no desenvolvimento de inovações digitais para enriquecer a vida das pessoas. Existem alguns aspetos que levam a duvidar se beneficia sequer aos utilizadores, tais como: sobrecarga de informação, violação de privacidade e falta de confiança na nova tecnologia.

Na formulação original da visão AmI foi atribuída com quatro elementos do sistema. Sendo eles:

- A consciência do contexto, pois o ambiente pode determinar o contexto de uma determinada atividade. Este contexto relaciona informações significativas entre as pessoas e o ambiente;
- A personalização, em que o ambiente pode adaptar-se às necessidades dos utilizadores, reconhecendo-os e reajustando as necessidades, de maneira a apoiá-los o máximo possível;
- Adaptativo, pois o ambiente pode adaptar-se às necessidades dos utilizadores, aprender com as situações recorrentes e reajustar-se consoante as necessidades de mudança;
- Antecipação, o ambiente deve agir ao utilizador sem meditação consciente. Sendo assim, torna-se capaz de captar características comportamentais e responder pré-ativamente aos utilizadores.

Estes elementos facilitam a comunicação nos ambientes AmI, fornecendo assim aos utilizadores a interação e o controlo inteligente. Contudo, devido às expectativas tecnológicas do AmI, a verdadeira inteligência do ambiente AmI requer a implementação de inteligência

social, isto significa a conformidade com as convenções sociais. Para este fim, os investigadores adicionam mais três elementos de inteligência social para os ambientes AmI [65].

Esses elementos podem ser formulados da maneira seguinte:

- Socializado, os conceitos de interação do utilizador aplicam os protocolos de comunicação que são compatíveis com convenções sociáveis, logo este segue as regras sociais e as éticas sociais;
- Empático, os conceitos de interação mostram a consciência do estado interno das emoções, os motivos do utilizador e a adaptação do estado do mesmo, demonstrando também a compreensão e a solidariedade;
- Consciente, o sistema tem um estado interno que expõe um comportamento transparente e consistente quando interage com as pessoas e é reconhecido pelo utilizador como algo consciente.

Os investigadores determinaram diferenças entre os sistemas e a inteligência social para nível conceptual, logo, estes devem ser integrados para o nível de implementação. Por exemplo, no ambiente deve ter noção quando deve respeitar a privacidade do utilizador num determinado contexto.

2.6.2.1.4. Outras perspetivas

Embora a visão do AmI tenha sido lançada há mais de 10 anos, o desenvolvimento, a implementação dos sistemas resultantes e os ambientes permanecem prematuros. Acontece que a maioria dos conceitos são disruptivos. Isto resulta, principalmente, devido à lacuna que existe entre a ficção dos conceitos e, por outro lado, a complexidade dos mesmos [65]. Existem sete tópicos que devemos distinguir para lidar com estas situações:

1. Controlo do ambiente

Como podemos aceder e controlar dispositivos num ambiente inteligente? O problema é como integrar o mundo físico num mundo digital. Isto diz respeito às funcionalidades tais como deteção, atuação, identificação marcada, assegurar acesso à rede sem fios e gestão de identidades. Um dos receios que nos leva a preocupar são as informações que são colecionadas nos objetos e nos arredores. Além disso, deve apoiar o contexto, o significado e a semântica. Finalmente, devemos apoiar o controlo de redes distribuídas para dispositivos inteligentes

(sensores e atuadores) que colecionam as informações do ambiente e pode gerar efeitos físicos através de dispositivos luminosos, altifalantes, ecrãs, etc.

2. Interfaces tangíveis

Quais são os conceitos de interação multimodal que os ambientes podem aplicar num ambiente inteligente? A questão é criar novos conceitos de interação que vão para além dos conceitos atuais de interface dos utilizadores, como metáfora do *desktop*, controlos remotos acionados através dos menus e interfaces de voz. O conceito destas interfaces tangíveis utiliza artefactos físicos como objetos para representação e interação. O objetivo principal é integrar entre os dois mundos: físico e digital. Os artefactos são frequentemente chamados como objetos tangíveis, apreensíveis ou hápticos.

3. Programação para utilizador final

Como os utilizadores finais conseguem programar as suas funcionalidades num ambiente inteligente? A questão é projetar ambientes de programação que permitam aos não especialistas em computação criar e modificar artefactos de *software* no ambiente inteligente. Portanto, este deve capacitar os utilizadores finais em adaptar os ambientes para as suas necessidades. Isto envolve métodos e ferramentas para editar, interpretar, executando e executar funções das aplicações de forma amigável e intuitiva.

4. Experiências sensoriais

Como as pessoas conseguem processar informação numa interação com os ambientes inteligentes? Devemos, então, perceber a interação entre o cérebro humano e o seu ambiente. Isto envolve elementos como modelar o processamento de informação do cérebro humano, simular a aprendizagem, aperfeiçoar o armazenamento da memória humana e colecionar de novo os processos. Também deve entender as emoções, portanto, o ambiente tem de criar essas experiências.

A questão-chave é como compreender o estado emocional dos utilizadores de forma confiável para estes ambientes. Capturar, influenciar e criar emoções são os novos campos para investigação para a computação afetiva.

As perguntas chave da investigação são:

- Como podemos modelar o estado de humor?
- Quais são os estados de humor que devemos captar?

- Como pode ativar um conjunto de sentidos de influência às emoções?

A deteção e a atuação dos efeitos sensoriais em relação com atividade humana são elementos importantes no desenvolvimento das soluções efetivas nas experiências sensoriais. O campo de deteção e atuação humana é devoto para este assunto e de forma que o estado que este campo permanece prematuro.

O receio que permanece na investigação de todos os tipos de parâmetros e efeitos podem influenciar emoções. As questões-chave da investigação são:

- Como podemos derivar descritores para emoções baseado na entrada de captura de emoção?
- Como podemos gear ou produzir efeitos emocionais apropriados?

As respostas para estas questões podem ser encontradas no domínio da pesquisa de perceção, que o campo de pesquisa na interseção da psicologia cognitiva e engenharia elétrica. O problema principal é como incorporar as emoções das pessoas na interação com conteúdo para resultar em experiências positivas? Para este caso precisa uma *framework* científica para captar, medir, julgar e explicar a experiência do utilizador.

5. Presença social

Como pode ser alcançado a interação social nos ambientes inteligentes? A questão é, como os ambientes podem apoiar os aspetos sociais da interação interpessoal. Na investigação, distinguem três elementos básicos numa presença social:

- A co-presença, o grau em que uma pessoa é vista como "real" na comunicação mediada. Por outras palavras, um ambiente acaba por ter uma consciência mútua sensorial do outro corporificado;
- O envolvimento psicológico, o nível de saliência da outra pessoa numa interação e resultante saliência do relacionamento interpessoal, ou seja, o acesso à inteligência da outra pessoa na interação;
- O compromisso comportamental, o grau do comportamento é sincronizado na interação entre atores mediadores. Por outras palavras, o comportamento exprimido durante a interação com a pessoa com o ambiente inteligente.

6. Persuasão confiável

Os ambientes inteligentes suportam o contexto sensível, fornecem conteúdo e interação persuasiva, de maneira confiável. De acordo com Fogg [68], podemos identificar as seguintes questões relacionadas com a persuasão:

- A redução substitui uma tarefa complexa por uma mais simples em virtude da introdução de automação e computação, como, também, a virtude de antecipação e definição das características do ambiente;
- A customização e a personalização ajustam as mensagens e o conteúdo às necessidades do indivíduo. Este requer um modelo rico e sensível que favoreça os utilizadores, ou seja, ir para além dos simples hábitos e das preferências. Isto, inclui dos aspetos das personalidades, dos estados de saúde, dos contextos, entre outros;
- A sugestão relembra as pessoas para realizarem determinados comportamentos em momentos oportunos. Propondo comportamentos então precisa de ser sensível para o contexto, isto torna-se num dos aspetos centrais do AmI;
- A auto monitorização permite que as pessoas se monitorizem e informa-as sobre como poderiam modificar os seus comportamentos. Isto acaba por ser muito tedioso e abre a oportunidade para o AmI facilitar o processo.

7. Inclusão e éticas.

O que é preciso que as pessoas aceitem que o ambiente está monitorizando todos os seus movimentos, esperando o momento certo para sumir o controlo com objetivo de cuidar os mesmos?

Grande parte dessa aceitação dependerá do seu benefício funcional dos ambientes e da disponibilidade de mecanismos que permitam aos participantes fazer as suas próprias escolhas de forma compreensível, transparente e independente do seu nível de compreensão.

A solução para este problema é frequentemente comunicada como ética embarcada que lidar com os direitos fundamentais ao nível de design dos sistemas embarcados correspondentes.

2.6.2.2. Topologias e Protocolos

Na parte da comunicação com os dispositivos, atualmente recorremos às tecnologias com ou sem fios (*wireless*), que são ativamente usados para estabelecer comunicações dentro da rede de uma casa ou de um edifício.

Como padrão para redes com fios, recorremos a Ethernet (*IEEE* 802.3), enquanto para as tecnologias sem fios recorremos a transmissão de dados. E quando recorremos muitos dispositivos ao mesmo tempo, reduzimos assim os custos, mesmo que sejam em zonas que são impossíveis de usar tecnologias com fios. As tecnologias *wireless* tornaram-se mais acessíveis para vários dispositivos e cada vez mais diminuiu o custo dos mesmos, contudo, permanece o problema da estabilidade e a eficácia de transmissão de dados [69].

Conforme exploramos as tecnologias *wireless*, especificamente para redes locais e pessoais (WLAN e WPAN), existe uma variedade de protocolos que podemos aplicar nas nossas casas. Na investigação [69], utilizaram um microchip ESP8266 numa rede TCP/IP e, durante a experiência, recorreram algumas topologias das quais foram: Ponto-a-Ponto (*Peer to peer*), Estrela (*Star*), Árvore (*Tree*) e *Mesh*. De forma a determinar as suas capacidades, recorreram aos seguintes protocolos: Bluetooth, ZigBee, Z-Wave, 6LoWPAN, Wi-Fi e RF [69].

Conforme estudaram estes protocolos notaram que o Wi-Fi e a ZigBee têm capacidade para todas as topologias existentes. Contudo, na investigação houve preferência da topologia *Mesh* para que todos os dispositivos se comunicassem entre si. Isto torna-se ideal para uma rede de uma casa inteligente, pois tem escalabilidade e estabilidade [69].

2.6.2.3. HTTP versus MQTT

Conforme a tecnologia evoluiu, houve maior abertura de escolha aos protocolos, como foi referido na secção 2.3, e juntando as aplicações IoT surge um novo protocolo padrão no quotidiano. No entanto, torna-se um desafio escolher um protocolo que seja eficiente e padrão, porque depende da natureza do sistema IoT e os seus requisitos das mensagens [70].

Normalmente, para aplicar essa comunicação numa rede IoT, compara-se os dois protocolos: o HTTP e o MQTT. O protocolo HTTP é predominantemente usado para as mensagens via *Web* [71, 72], isto tornou-se num protocolo ideal. Contudo, usando o HTTP para IoT acaba por não ser fiável, estes são os seguintes motivos [72]:

1. Comunicação de um para um: Na parte da comunicação, o protocolo HTTP está desenhado para comunicar entre dois sistemas e funciona numa página *Web* ou semelhante. Para IoT não se torna ideal usar esta comunicação, pois as aplicações IoT utilizam múltiplos dispositivos e estes são geram múltiplos dados;

2. Unidirecional: O protocolo HTTP é considerado unidirecional, ou seja, apenas permite enviar dados numa só direção. Para as aplicações IoT é preciso enviar os dados em ambas as direções, simultaneamente;
3. Sincronização: Nesta parte o cliente tem de esperar que o servidor responda. O problema da sincronização faz com que ocorra bloqueios num sistema, como por exemplo: as tarefas e os ciclos de CPU. Portanto, tornar-se-ia mais lento para as aplicações IoT, pois os sensores têm poucos recursos e consequentemente não conseguem trabalhar de forma síncrona. As alternativas seria usar os protocolos assíncronos tais como por exemplo: o MQTT, o Z-Wave e o ZigBee;
4. Escalabilidade: As ligações HTTP utilizam os recursos do sistema, especialmente nas tarefas Input/Output (I/O). Para todas as ligações, o cliente/servidor precisa de abrir uma ligação TCP persistente e subjacente. Cada vez que são adicionados os sensores na rede, a carga no servidor aumenta e se os sensores estão ligados em múltiplos dispositivos, aumenta a carga num pequeno sistema;
5. Alto consumo de energia: Tal como foi apontado anterior, este protocolo usa muitos recursos o que faz com que haja maior consumo de energia.

Depois de tomarmos conhecimento sobre as limitações do protocolo HTTP, tivemos que explorar protocolos que são mais eficazes num sistema IoT [72].

Um dos protocolos que estivemos a explorar foi o protocolo MQTT (*Message Queuing Telemetry Transport*). Este recorre a uma arquitetura "*publish-subscriber*" [71, 72, 73, 74], que fornece uma troca de mensagens entre o *publisher* e o *subscriber* que estão dependentes do *broker*, o servidor que faz com que transmitam as mensagens do *publisher* para os *subscribers* e vice-versa.

O protocolo MQTT [74] usa um protocolo de mensagens OASIS que não requer muitos recursos e que pode ser utilizado nos microcontroladores. Estes dispositivos possibilitam o envio de múltiplas mensagens, de forma assíncrona, das quais contêm informações importantes que pretendemos trocar.

Um conceito importante deste protocolo são os *topics* (tópicos). Isto permite que os clientes do MQTT partilhem as suas informações. Estas são estruturadas por uma hierarquia semelhante às pastas e ficheiros, usando uma barra diagonal ("/"), como um delimitador. Os nomes dos *topics* são *case sensitive*, usam uma codificação UTF-8 e deve consistir pelo menos um caracter para validar um *topic* [75].

Em termos de comparação dos protocolos, os investigadores [73] exploraram as funcionalidades dos dois protocolos recorrendo a um microcontrolador. Resumidamente, notaram que a consistência do *request-response* no protocolo HTTP causa um grande consumo na rede. Enquanto que no protocolo MQTT notaram que facilita o uso de múltiplos dispositivos, graças à arquitetura *publish-subscribe*.

2.6.2.4. Protocolo MQTT com Microcontrolador ESP8266

O estudo [76] investiga o funcionamento dos sensores que se integram com os aparelhos usando uma ligação *wireless* de baixo consumo. Explorando o protocolo MQTT com o microcontrolador ESP8266 do fabricante chinês Espressif Systems, pode ser visto como um "Arduino com capacidade de conectividade *wireless*" e que "pode comunicar com o servidor *wirelessly*" [47].

Este microcontrolador contém um sensor LDR que deteta a intensidade da luz, esse torna-se num cliente do MQTT e publica as informações para o MQTT *broker* e subscreve os comandos devido a atuação. O LED e o *buzzer* são usados como atuadores no protótipo. O módulo do ESP8266 publica os dados sob o tópico "esp\sense" e subscreve para os tópicos "esp\led" e "esp\buzzer" para receber os comandos que controla os dois atuadores. Outros clientes MQTT podem recorrer a outros dispositivos para comunicar com o servidor MQTT através da tecnologia existente Ethernet, WiFi, rede móvel e entre outros, tal como vemos na Figura 6.

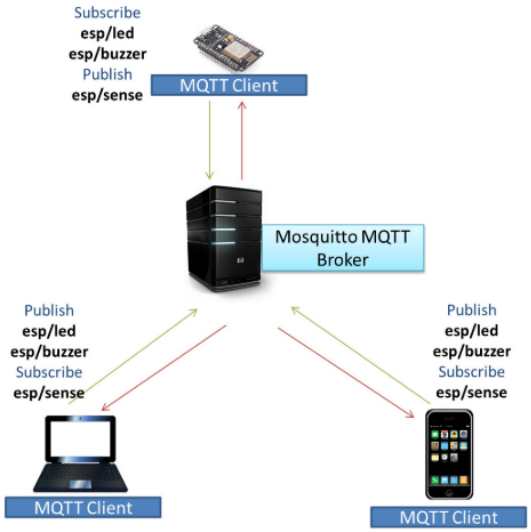

Figura 6. Transmissão de mensagens com MQTT [76].

O servidor MQTT utiliza o Eclipse Mosquitto, é um *message broker open-source* que controla as mensagens para um ou mais dispositivos, pode ser usado para computadores de

baixa potência (*Barebone*), como também, pode ser usado para computadores potentes (*Workstation*) e é compatível para muitos sistemas operativos.

Os investigadores fizeram uma experiência com uma aplicação base de Android que fizeram "MQTTLens" e "MyMQTT", onde os múltiplos atuadores podiam interagir tudo ao mesmo tempo.

Concluíram que o protocolo facilita a comunicação exigindo pequeno consumo de energia e pouca largura de banda.

2.6.2.5. Conectividade com MQTT usando Home Assistant e OpenHAB

A investigação [47] recorre a uma comparação entre os dois sistemas operativos *open-source*: OpenHab e Home Assistant. Implementaram esses dois sistemas numa rede IoT, a base de sensores que deteta a temperatura, a humidade e os movimentos. Os investigadores utilizaram um microcontrolador ESP com os sensores incluídos e adicionaram um Raspberry Pi para obter as informações por parte do microcontrolador. Para obter essas informações, tiveram que implementar um *broker* MQTT para estabelecer a comunicação entre os eletrodomésticos e o Raspberry Pi.

Os investigadores instalaram esses dois sistemas no Raspberry Pi e compararam as suas funcionalidades. Estes sistemas são essenciais quando os utilizadores querem desenvolver um sistema para uma casa inteligente.

2.6.2.5.1. OpenHab

O sistema OpenHab [46, 47] recorre a linguagem de programação Java. Tem uma documentação completa e com qualidade, devido à sua maturidade de desenvolvimento e aos numerosos *developers*. Torna-se num *software* adaptável para novos utilizadores. Este *software* pode ser instalado em diversos sistemas operativos: Linux, Windows e macOS.

Este sistema contém muitas extensões para protocolos de comunicação (Bluetooth, Z-Wave, ZigBee, entre outros), para dispositivos inteligentes (Philips Hue, Xiaomi Mi Smart Home, entre outros) e para serviços *Web* (REST API, VPN, IFTTT).

Atualmente, tem uma comunidade relativamente estagnada, pois os utilizadores e os *developers* passaram para novos projetos [47]. Se os novos utilizadores tiverem dificuldades podem recorrer a documentação, no entanto, se for preciso uma ajuda mais específica, existe o fórum da comunidade para resolver problemas específicos.

Este sistema contém uma ligação remota através do sistema que fornece uma gestão completa e que não força o utilizador a expor o servidor na Internet. Fornece, ainda, ao utilizador a possibilidade de aceder outros serviços como *cloud* e à integração de assistentes de voz: o Google Assistant e a Alexa.

2.6.2.5.2. Home Assistant

O sistema Home Assistant [45] recorre à linguagem de programação Python 3 e pode ser instalado em qualquer sistema operativo existente, de forma mais detalhada como no capítulo seguinte.

O Home Assistant contém muitas extensões para vários fins, tal como acontece no OpenHab, também pode ser implementado outros programas tais como Visual Studio Code, Spotify, AppDaemon (para recorrer a programação de Python) e entre outros.

O problema que existe no sistema Home Assistant é a documentação oficial [44]. Os novos utilizadores notarão alguma dificuldade para configurar e interagir com o sistema. Contudo, o Home Assistant tem uma comunidade ativa e que continua a crescer, portanto, se os utilizadores tiverem dificuldades a comunidade consegue responder.

A parte da ligação remota tem uma gestão limitada e exige uma subscrição para recorrer aos assistentes de voz Google Assistant e Amazon Alexa. Se for para aceder remotamente ao sistema acaba por ficar exposto na Internet tornando-o mais vulnerável.

2.6.2.5.3. Comparação dos sistemas

Os investigadores [47] exploraram a gestão remota e a configuração para os dois sistemas mencionados anteriormente. Notaram que ambos os sistemas seguem os objetivos de facilitar os utilizadores e integrar diferentes eletrodomésticos num só sistema.

A única incerteza para os investigadores acaba por ser a linguagem de programação para um novo utilizador. O OpenHab exige alguma base de conhecimento com Java e deve perceber o conceito da Programação Orientada por Objetos (POO). No Home Assistant a linguagem de programação de Python é mais acessível.

Ao concluírem a investigação, os investigadores propuseram a pergunta seguinte: “Qual destes é o melhor?”. A resposta depende dos objetivos que o utilizador deseja usar nestes sistemas. Se for um utilizador inexperiente em programação seria recomendável usar o Home Assistant, mas se o utilizador tiver algum conhecimento de programação torna-se recomendável o OpenHab.

2.6.2.6. Outras investigações na UMa

Com o decorrer da investigação sobre sistemas operativos para casas inteligentes, foram aplicadas investigações na unidade curricular de Sistemas Personificados e Ubíquos, precedentemente chamado Computação Ubíqua, que motivaram a pesquisa dos sistemas de OpenHab e de Home Assistant, fornecendo, também, as opiniões e as conclusões dos mesmos.

2.6.2.6.1. Smartbox: Caixa de correio inteligente

A investigação [48] desenvolveu uma caixa de correio inteligente, ou *smartbox*, com o objetivo de fortalecer segurança para os utilizadores que recebem as encomendas, pois devido à situação atual da pandemia acabou por motivar as compras *online*. O problema que passa a ocorrer são as encomendas de maior de dimensão que acaba por aumentar o montante da encomenda.

Esta caixa de correio segue a um conjunto de benefícios como a comodidade, a redução dos custos de entrega, a proteção contra roubo de encomendas, a autenticação remota, o local e o distanciamento social.

Nesta caixa foram instalados os seguintes *hardwares*: a fechadura eletrónica, a Raspberry Pi 4, o sensor RFID RC522 e o sensor de pressão. Foi adicionado, ainda, um código QR que representa a identificação da caixa.

No Raspberry Pi 4 utilizaram o sistema Home Assistant para gerir o funcionamento *smart* da caixa e gerir, também, os dados para a aplicação móvel Home Assistant, de maneira a controlar e a interagir remotamente o sistema.

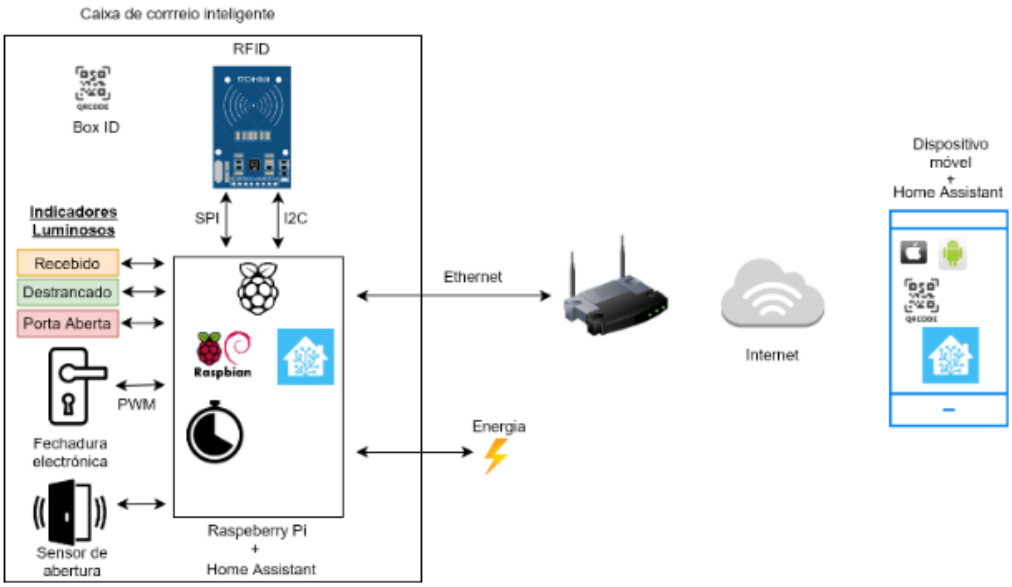

Figura 7. Arquitetura da Smartbox [48]

Como observamos na Figura 7, na caixa implementaram o sistema Raspbian, uma distribuição Linux base Debian, com o Home Assistant incluído. O RFID faz a identificação e a autenticação dos cartões que interagem com a caixa. Nesta caixa contém LEDs para notificar o utilizador quando interage com a caixa. A luz laranja indica que foi recebido o pedido, a luz verde indica que foi destrancado e a luz vermelha indica que a porta está aberta.

Instalaram a versão Supervised do Home Assistant, em vez da versão Operating System, devido às limitações que tinham para aplicar *script*s. Com a versão Supervised conseguiram aplicar modificações através dos *add-ons* existentes do *software*.

Outra funcionalidade que implementaram foi caso ocorresse alguma falha devido à tomada de alimentação ou caso o Raspberry teve de reiniciar de forma inesperada. Outras modificações que adicionaram para a caixa foram as automatizações do sistema Home Assistant para garantir o funcionamento do sistema inteligente.

Concluíram que o sistema tem imensas vantagens para interagir com múltiplos dispositivos, contudo, apontam que ambos os sistemas "têm um caminho a percorrer para se tornarem intuitivas e mais 'amigas' do utilizador", devido à grande quantia de informação, de funcionalidades e de documentação. Outros pontos que notaram foram as instruções da linguagem de programação, são pouco claros e contêm tutoriais incompletos ou mal estruturados.

2.6.2.6.2. Home Assistant para Gestão de Casas Inteligentes

Nesta investigação [49], os alunos planearam um sistema de irrigação automática com diversos sensores (medição de temperatura, medição da qualidade do ar e medição da humidade), tentaram explorar alguns exemplos de irrigações automáticas e apenas encontraram um trabalho relacionado.

Nesse trabalho relacionado referem um sistema de irrigação automática de uma planta usando o Arduino UNO com um sensor de humidade. Se o solo ficasse seco, ativava o sistema de rega até que o solo ficasse suficientemente molhado.

Implementaram um Raspberry Pi com Home Assistant e um Arduino UNO com sensores e atuadores. Dentro do sistema configuraram no ficheiro de configuração do Home Assistant para estabelecer ligação entre os sensores e no Arduino atribuíram o mapeamento dos valores lidos no Arduino de forma que sejam bem interpretadas para o utilizador final com um formato visual *user friendly*.

De seguida, adicionaram a aplicação móvel do Home Assistant para estabelecer controlo com o sistema, posteriormente, implementaram automatizações no Home Assistant. Quando os sensores detetam o movimento, aciona um dos LEDs para notificar e simular a segurança do sistema. Adicionaram também as automatizações para enviar ao utilizador as informações sobre o estado da bomba de água.

Para finalizar o projeto, cada elemento do grupo personalizou *dashboards* para que conseguissem explorar facilmente e fornecer a possibilidade de customizar os cenários para o utilizador final. Acabaram por obter bons resultados conforme esperados.

Indicaram que tem uma "vasta documentação" que faz com que os utilizadores experientes se integram ao desenvolvimento do Home Assistant.

Concluíram que no desenvolvimento do projeto foram completados com sucesso e determinou um melhor conhecimento e interesse às tecnologias IoT ao grupo.

2.6.2.6.3. Gestão de Tomadas Inteligentes

A investigação [50] consiste em perceber o funcionamento quando interagem entre o OpenHab e as tomadas inteligentes TP-Link Tapo P100. Os objetivos do projeto tendiam controlar remotamente e definindo um horário próprio para ativar os eletrodomésticos e minimizando os custos energéticos.

Instalaram num Raspberry Pi com openHabian que instala o sistema do OpenHab. Ao explorarem a aplicação do OpenHab tiveram um imprevisto, as tomadas adquiridas pertenciam

a uma companhia separada da TP-Link e não são suportadas no OpenHab. Portanto, tiveram que implementar uma biblioteca que fosse compatível para implementar no sistema.

Após a instalação das tomadas tiveram que configurar um painel que permitisse ligar e desligar as tomadas remotamente, contudo, notaram que foi uma configuração complexa para integrar no painel. Outras tentativas que recorreram foi com assistentes de voz, mas não houve sucesso.

Concluíram que as metodologias aplicadas, não conseguiram efetuar compatibilidade entre as tomadas e o OpenHab devido as dificuldades que apontaram na investigação. Ainda apontaram que o OpenHab não se torna ideal para novos utilizadores por ser pouco intuitivo e complexo.

2.6.2.7. Comparações entre Hubitat e SmartThings

Como foi referido na secção 2.5.4, o Hubitat e o SmartThings acabam por ter algumas semelhanças.

O sistema SmartThings apareceu no mercado há algum tempo. Tem alguma maturidade em termos de compatibilidade. A terceira geração do SmartThings e pode ser ligado na rede via local (LAN) ou *wireless*. Para o sistema Hubitat entrou recentemente no mercado, por isso, não tem muita compatibilidade de dispositivos e podemos ligar na rede apenas localmente [54, 55].

Ambos os sistemas SmartThings e Hubitat, são fáceis de configurar os dispositivos inteligentes, utilizam os protocolos de comunicação Zigbee e Z-Wave e ambos conseguem recorrer aos assistentes de voz (por exemplo: o Google Assistant e a Alexa) [54, 55].

Existem algumas diferenças que separam o funcionamento destes dois sistemas e, como também, motiva a preferência dos utilizadores entre automatização, privacidade e segurança.

O SmartThings está completamente dependente da base Internet, o sistema não fornece liberdade ao utilizador de escolher as atualizações do sistema e não tem forma de customizar uma automatização do sistema. Na parte da segurança este sistema torna-se vulnerável aos ataques de terceiros [54, 55].

Quanto ao Hubitat é completamente local. Se a Internet cair ou deixar de funcionar, o sistema permanece intacto e seguro. O utilizador pode escolher as atualizações do sistema que deseja e podemos customizar as automatizações [54, 55].

2.6.2.8. *Open Source* versus *Closed Source*

O mercado teve sempre uma divisão de interesse entre dois tipos de *software*: *open source* e *closed source*.

O *software open source* recorre com código fonte aberta que podemos partilhar, modificar e colaborar com os *developers* e ao resto da comunidade mundial.

Para o *software closed source*, o desenvolvimento de *software* torna-se num processo especializado, controlado por uma equipa especializada que trata o projeto com os cuidados possíveis e, ocasionalmente, aplicam melhorias para as novas versões [77].

Resumidamente, *open source* representa um "estilo bazar" e o *closed source* como um "estilo catedral" [77, 78].

2.6.2.9. Windows XPe e Windows IoT

Na empresa Microsoft, na sua família da Windows apesar de serem sistemas *closed source*, também fornecem sistemas especializados para um *hardware* mais limitado. Desde o lançamento do Windows XP, lançaram a versão Windows XP Embedded (abreviadamente como XPe) [79].

Atualmente, muitos desses sistemas permanecem em muitas caixas de multibanco (ATM). Conforme passou o tempo aparecia outras versões, mas não aparecia um sistema apto e cada vez mais houve a falta de suporte por parte do Windows CE (também conhecido como *Embedded Compact*). Portanto, os sistemas *open source* começaram a dominar nos sistemas embebidos e causou receios à Microsoft [79, 80].

Recentemente, a Microsoft aplicou mudanças graças ao lançamento do novo sistema operativo Windows 10. Isto mudou o rumo do desenvolvimento e da execução em muitos dispositivos, especialmente em IoT [79]. Portanto, criaram o sistema Windows 10 IoT, a evolução do Windows Embedded [79].

Existem atualmente quatro versões IoT: o Windows 10 IoT Core, o Windows 10 IoT Core Services, o Windows IoT Enterprise e o Windows Server IoT. São sistemas que estão especializados para executar em dispositivos de menor dimensão que utilizam arquiteturas ARM e x86-64 e ocupam uma 'pegada' muito menor de armazenamento, ao contrário do resto da família Windows 10 e Windows 11. As versões IoT Enterprise e Server IoT, utilizam sistemas equivalentes da família Windows 10 Enterprise e Windows Server 2022, a versão Enterprise é licenciada para dispositivos dedicados para soluções IoT e a versão Server está especializado para a gestão e segurança para as soluções do IoT.

Os sistemas Windows IoT são utilizados para um conjunto de cenários, tais como por exemplo: os edifícios inteligentes, as assinaturas digitais, os ATM, os POS, os sistemas de vídeo vigilância, a automação industrial e as redes de telecomunicação.

3. Desenvolvimento

Neste capítulo analisamos os requisitos para implementar os dois sistemas de casas inteligentes, o sistema Home Assistant e Hubitat, mostrando as decisões e justificações às implementações que foram feitas. Desta forma, é explorado, resumidamente, as arquiteturas e outras opções que podem ser aplicadas nos dois sistemas.

3.1. *Hardwares* utilizados

No desenvolvimento desta tese, foram exploradas as funcionalidades dos dois sistemas com alguns dispositivos inteligentes e sensores. No decorrer deste texto, iremos separar em duas partes a apresentação dos dois sistemas e, no fim, iremos abordar o *hardware* que foi compatível para ambos os sistemas.

3.1.1. *Hardwares* utilizados no Home Assistant

- Raspberry Pi 3 Model B+

Este modelo do Raspberry foi utilizado para iniciar a investigação e implementar o sistema operativo do Home Assistant. Como foi referido anteriormente, no capítulo anterior, o sistema Home Assistant pode ser implementado em qualquer tipo de computador, mas, se usarmos um computador de placa única, facilita a instalação e motiva o conceito minimalista.

Na parte dos requisitos para o Home Assistant, este Raspberry Pi tem recursos suficientes como o Cortex-A53 (ARMv8) 1.4GHz e um 1GB LPDDR2. Em termos de armazenamento foi utilizado um cartão SD de 16 GB, o que não é recomendável, mas foi o suficiente para a realização dos testes.

- Microcontrolador ESP8266 NodeMCU

Este microcontrolador é um modulo Wi-Fi de baixo consumo com o protocolo TCP/IP, fabricado pela empresa Espressif Systems [82]. Nos últimos anos, o microcontrolador tornou-se muito comum para o contexto IoT e para estabelecer comunicação com outros dispositivos que estão ligados ao Arduino ou a outro dispositivo de uma casa inteligente, tal como mencionado na secção 2.6.2.4, que foram usados no sistema Home Assistant [11].

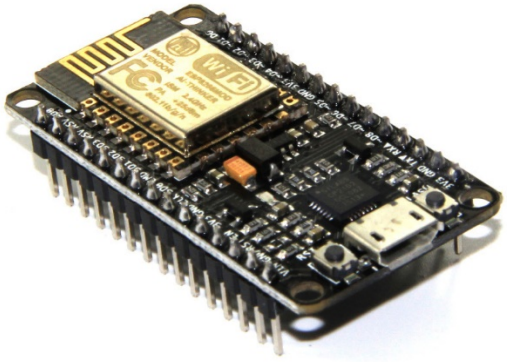

Figura 8. ESP8266 NodeMCU [82]

A empresa Espressif Systems, no ano 2014, desenvolveu um SDK (*Software Development Kit*) para programar com este microcontrolador e, desde então, começou a aparecer mais outros SDK *open-source* como: o Arduino, o ESP Easy, o ESPHome, o NodeMCU e o MicroPython, entre outros. Cada SDK existente recorre a uma determinada linguagem de programação. Todos estes SDK tornaram-se ideais para desenvolvedores, investigadores e entusiastas para casas inteligentes. Nesta tese, usamos a extensão do Home Assistant ESPHome [82, 83].

- Sensores

Para iniciar os primeiros testes, foram adicionados os sensores AM2302 (DHT22) para detetar a temperatura e a humidade do quarto.

- SONOFF Universal Zigbee

Este dispositivo [83] serve para implementar facilmente o protocolo Zigbee em qualquer computador. Pode servir como um *gateway* de Zigbee para vários sistemas operativos *open source* para casas inteligentes, especialmente para o Home Assistant.

Basta só instalar a integração ou extensão que seja compatível com SONOFF. Outra vantagem, não precisa de investir Hubs Zigbee de marcas dispendiosas, como por exemplo Philips Hue Bridge [84].

Torna-se num dos dispositivos que tem maior compatibilidade com outros dispositivos inteligentes e tem um bom alcance para redes extensas. Opcionalmente, podemos atualizar o *firmware* para melhorar as funcionalidades do aparelho.

- Texas Instruments CC2531

Foi o segundo dispositivo Zigbee [85], que foi testado para comparar o dispositivo que foi mencionado anteriormente. Pela documentação, é um dispositivo desatualizado e com menor capacidade de alcance.

3.1.2. *Hardware* utilizado no Hubitat

- Hubitat Elevation

Este modelo do Hubitat tem funcionalidades pré-implementadas. Por outras palavras, ao contrário do Raspberry Pi, tem protocolos implementados para estabelecer ligação com outros dispositivos inteligentes compatíveis com os protocolos Zigbee e Z-Wave. Também não exige muita configuração para ativar os aparelhos.

3.1.3. *Hardware* compatível com os dois sistemas

Os dois sistemas Home Assistant e Hubitat, tem um conjunto de dispositivos que são compatíveis e podemos usar para ambos estes sistemas graças aos protocolos existentes que efetuam comunicação com os dispositivos.

Neste projeto foram usados os dispositivos Philips Hue. Este é um dos fabricantes [86] mais conhecidos para implementar luzes com ou sem funcionalidades inteligentes, como foi referido anteriormente no capítulo anterior. As funcionalidades destes fabricantes centram em luzes inteligentes a qualquer ponto da casa, pode ser interagido pelo utilizador ou através da deteção por um sensor.

Figura 9. Philips Hue Bridge

- Lâmpadas GU10

Nesta experiência foram utilizadas as lâmpadas GU10 que podemos manipular a intensidade e a temperatura da luz. Também foi utilizado a Hue Bridge (Figura 9) que serve para expandir as funcionalidades dos dispositivos que estão presentes na rede da casa, sem interagir com o protocolo Bluetooth.

- Philips Hue Bridge

O Philips Hue Bridge usa o padrão Zigbee Light Link, de acordo com a Philips, tem compatibilidade com todas as lâmpadas e dispositivos que se torna claro que são compatíveis com os aparelhos da mesma marca, também tem alternativas para outros fabricantes [87].

3.2. Home Assistant

Como a plataforma do Home Assistant (HA) é *open source*, tem várias formas para ser instalada em qualquer dispositivo, tal como foi referido na secção 2.6.2.5.2. Mais detalhadamente, a aplicação pode ser instalada de várias formas:

- Num sistema Linux Debian ou que usa base Debian (Raspbian, Ubuntu, entre outros);
- Num sistema do Windows 10, com ou sem WSL (Windows Subsystem Linux);
- Num sistema MacOS, através de uma máquina virtual (VirtualBox ou KVM);
- Numa máquina virtual, recorrendo VMWare, VirtualBox ou KVM (Kernel-based Virtual Machine) se estiver num sistema Unix ou Unix-*like*;
- Num contentor através do Docker;
- Num Raspberry Pi ou num ODROID ou num Intel NUC ou semelhante com o Home Assistant OS.

A maioria das opções que foram apresentadas recorrem a uma configuração manual que exige alguns requisitos de pacotes, entre outros. O último método de instalação, a versão Home Assistant OS, apenas exige configurar o endereço de rede. Os *developers* recomendam esta versão do Home Assistant, especialmente para utilizadores que não tenham experiência de programação.

Conforme explorados nas próximas secções, estas apontam como funciona a arquitetura do sistema, que leva a explorar as camadas do Home Assistant, as integrações e os API. Todas estas informações existem na página dos *developers* do Home Assistant [88]. Também, contém informações essenciais, caso estejamos interessados em implementar funcionalidades para interagir com o Home Assistant. Inicialmente, o utilizador tem que ter algum conhecimento de programação com a linguagem de programação Python e, essencialmente, com a linguagem de programação YAML para configurar o sistema.

3.2.1. Arquitetura do Home Assistant

Na arquitetura deste sistema é fornecido uma plataforma que permite o controlo doméstico e a automação residencial, no entanto, Home Assistant é um sistema embebido que fornece a mesma experiência tal como acontece com qualquer outro produto que fica pronto para o seu uso. A integração, a configuração e a atualização são todas feitas para uma interface fácil de utilizar.

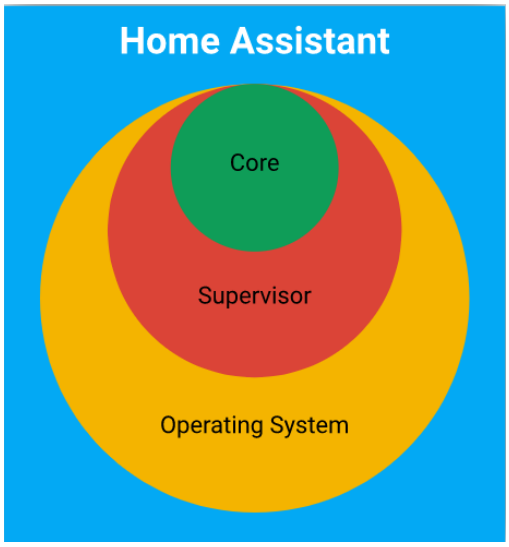

Figura 10. Arquitetura do Home Assistant [88].

Este sistema está separado em três camadas [88]:

- Sistema operativo, que fornece um ambiente Linux mínimo para executar as duas camadas inferiores;
- *Supervisor*, que faz a gestão do sistema operativo;
- *Core*, que interage com o utilizador, o supervisor, os dispositivos e os serviços da IoT.

Estas camadas representam o conceito e como funciona o sistema do Home Assistant. Contudo, os utilizadores têm opções sobre especifícas camadas. Os utilizadores podem recorrer às três versões: Home Assistant OS (*Operating System*), Home Assistant Supervised e Home Assistant Core [89].

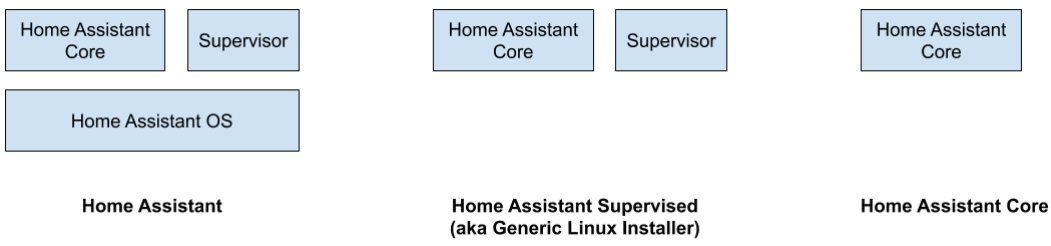

Figura 11. Versões de Home Assistant [89].

Home Assistant Core é um programa de Python que podemos executar num ambiente virtual, em qualquer sistema operativo e podemos recorrer como uma imagem Docker. Ao instalarmos esta versão do sistema, apenas temos a camada principal com as integrações, ou seja, não fornece o painel Supervisor nem temos possibilidade de adicionar ou criar extensões [88, 89].

Home Assistant Supervised (também conhecido como Home Assistant on Generic Linux) fornece a completa experiência do Home Assistant num sistema Linux (numa máquina virtual ou num *bare metal*) que recorre a uma base Debian (Ubuntu, Raspibian, entre outros). A única diferença entre Home Assistant OS e Home Assistant Supervised é que não tem a possibilidade de atualizar na interface do Home Assistant. Mesmo que o utilizador tenha completo controlo do Home Assistant, é preciso ter algum conhecimento de sistemas Linux [88, 89, 90].

Home Assistant OS contém todas as três camadas do sistema, tem todas as funcionalidades das versões anteriores e tem um gestor de versões do sistema. O sistema está otimizado para computadores de placa única, tais como Raspberry Pi, ODROID, Intel NUC e Tinker Board. Não se baseia num sistema regular de Linux. [88, 89].

3.2.2. Arquitetura do Núcleo

A arquitetura do Home Assistant Core (Figura 12), contém quatro partes principais:

- *Event Bus* – facilita o disparo e escuta de eventos;
- *State Machine* – acompanha os estados das coisas e dispara o evento "*state_change*" quando o estado mudou;
- *Service Registry* – escuta os eventos para os eventos "*call_service*" e permite outros códigos para registar serviços;
- *Timer* – envia ao evento "*time_changer*" cada segundo no *Event Bus*.

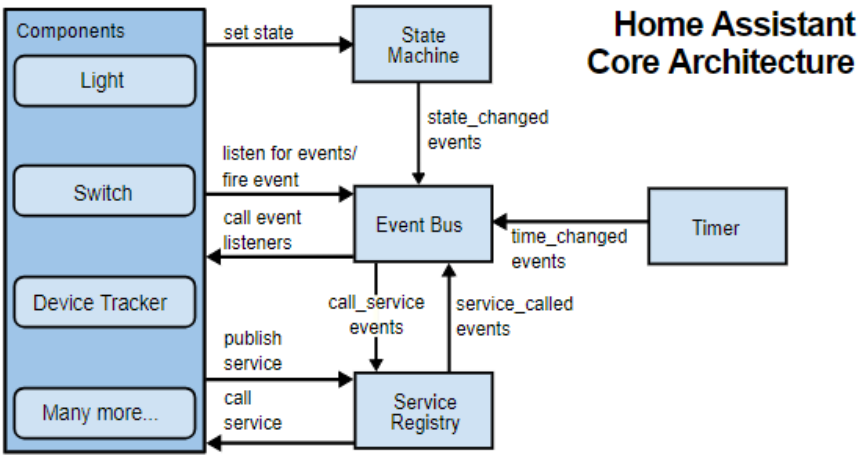

Figura 12. Arquitetura do Núcleo [88]

3.2.3. Integração da arquitetura

O Home Assistant Core pode ser estendido com integrações e cada integração fica responsável por um domínio específico no Home Assistant. As integrações podem escutar e atuar em certos eventos, a oferecer serviços e manter os estados [88].

Todas as integrações recorrem por componentes e as plataformas fazem com que se integrem com outras integrações. Estas integrações foram programadas através de Python, portanto, podemos aplicar qualquer operação possível que Python consiga oferecer. Atualmente, Home Assistant oferece uma grande quantidade de integrações [88].

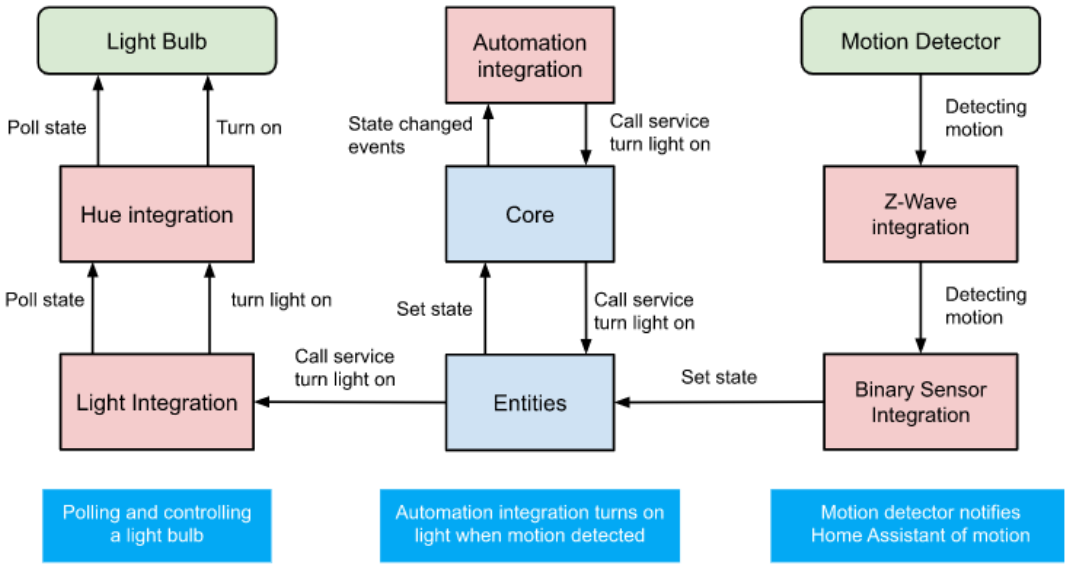

Figura 13. Integração do Home Assistant [88].

A Figura 13 apresenta Home Assistant que consegue distinguir as integrações de Z-Wave e de Philips Hue. Estas integrações executam-se quando aplicamos uma ação, ou seja,

quando acionamos um serviço de ligar ou desligar a luz, quando ocorre uma mudança de estado ou quando deteta algum movimento [88].

Com estas integrações conseguimos definir a categoria de dispositivos IoT no Home Assistant, sendo que, neste caso, é para as luzes. Portanto, a integração da luz fica encarregue de fornecer os dados e determinar o formato dos mesmos para o sistema do Home Assistant [88].

As integrações conseguem interagir com os dispositivos e os serviços externos e tornam-se disponíveis no Home Assistant [88].

Existem pequenos pedaços de lógica de automação residencial que aplicam tarefas comuns numa casa inteligente. A integração mais popular é a "*automation integration*" que permite aos utilizadores criarem automações através de um formato de configuração. Também existe a integração "*flux integration*" que controla as luzes através da posição do sol [88].

3.2.4. Autenticação

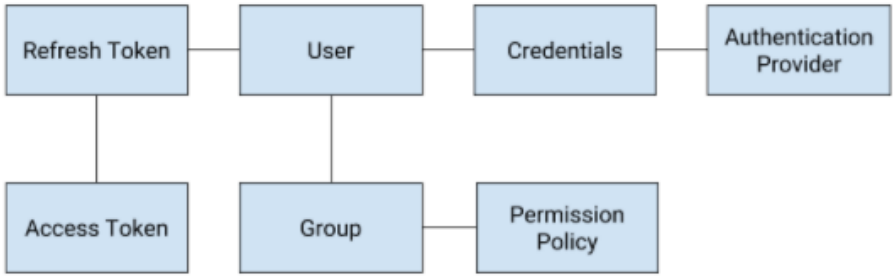

Figura 14. Autenticação do sistema [88]

O sistema do Home Assistant contém um sistema de autenticação [88] para que diferentes utilizadores interajam com o mesmo. Este sistema de autenticação, como mencionado na Figura 14, consiste em várias partes:

O fornecedor de autenticação, disponibiliza aos utilizadores a autenticação e está dependente do próprio fornecedor, que escolhe o método de autenticação. Como padrão, podemos ativar o fornecedor que armazena os utilizadores com segurança dentro do diretório de configuração.

As credenciais armazenam a autenticação do utilizador com um fornecedor específico de autenticação. Permitem que encontremos o utilizador no sistema e, se este não existir, será logo criado. Contudo, só fica ativo se for aprovado pelo dono do sistema.

Em relação aos utilizadores, cada pessoa é representada como um utilizador do sistema. Quando o utilizador efetua o acesso, recebe uma *token* para aplicar os pedidos no Home

Assistant. Existe um utilizador declarado como dono do sistema que configura e autoriza as permissões para os outros utilizadores.

Os grupos fazem com que se permita o acesso a um conjunto de utilizadores.

As políticas de permissão descrevem quais são os recursos que um grupo tem acesso.

O acesso e a atualização do *token* acontece quando uma aplicação quer ter acesso ao Home Assistant. É necessário perguntar primeiro ao utilizador para começar a execução do mesmo. Nisto, resulta um código de autorização para estas aplicações. Este código pode ser usado para receber um acesso e atualizar uma *token*. O acesso a uma *token* torna-se temporário e irá permanecer válido enquanto acontecer o refrescar das *token*. Deixa de existir quando o utilizador elimina essa mesma *token*.

As permissões são recursos experimentais que não estão ativados. Estas limitam quando um utilizador pretende ter acesso ou controlo. Também estão incluídas nos grupos que o utilizador pode fazer parte. As permissões determinam o controlo de acesso que podem ser aplicados para os membros. Contudo, não se aplicam aos utilizadores que são considerados como donos, pois estes têm acesso a todas as funcionalidades.

Cada utilizador tem a sua instância que fornece a cada um deles o controlo dos seus próprios dados. Os *developers* adotaram OAuth2 para este API e combinaram com a extensão OAuth2 IndieAuth.

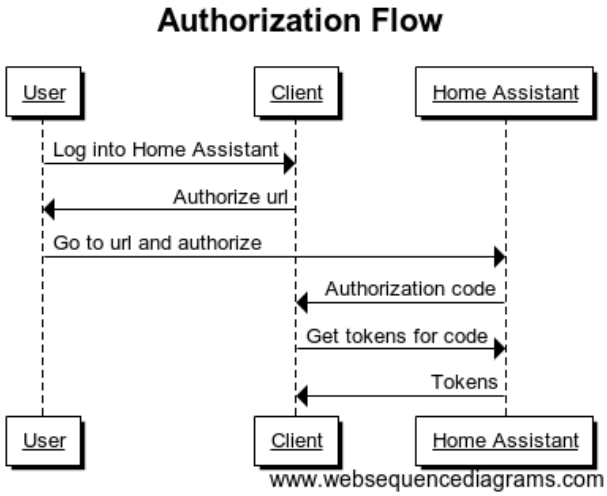

Figura 15. Fluxo de Autorização [88]

3.2.4.1. REST API

O Home Assistant fornece um RESTful API [88] na porta do *front-end* da página *Web*, porta pré-definida 8123. Se não for usado na configuração, tem de ser adicionada a componente do API no ficheiro de configuração.

Para devolver os objetos codificados JSON do RESTful API devemos utilizar o endereço do Home Assistant com "api" (http://IP_ADDRESS:8123/api). Existem muitas formas para consumir o REST API do Home Assistant. Podemos aplicar com as seguintes opções:

- Com cURL

```
curl -X GET \
  -H "Authorization: Bearer ABCDEFGH" \
  -H "Content-Type: application/json" \
  http://IP_ADDRESS:8123/ENDPOINT
```

Figura 16. Opção através do cURL [88]

- Com Python

```
from requests import get

url = "http://localhost:8123/ENDPOINT"
headers = {
    "Authorization": "Bearer ABCDEFGH",
    "content-type": "application/json",
}

response = get(url, headers=headers)
print(response.text)
```

Figura 17. Opção através do Python [88]

Também podemos recorrer à integração *Restful Command* numa automação ou *script* do HA.

```
turn_light_on:
  url: http://localhost:8123/api/states/light.study_light
  method: POST
  headers:
    authorization: 'Bearer ABCDEFGH'
    content-type: 'application/json'
  payload: '{"state":"on"}'
```

Figura 18. Opção do Restful Command [88]

As chamadas com sucesso podem retornar com código 200 ou 201. Outros resultados podem devolver os seguintes códigos:

- 400 (*Bad Request*);
- 401 (Não autorizado);
- 404 (Não Encontrado);

- 405 (Método não permitido).

3.2.4.2. WebSocket API

O Home Assistant contém um WebSocket API que pode ser usado para transmitir informações de uma instância deste sistema, para qualquer cliente que implementa WebSocket. As implementações podem recorrer com diferentes linguagens de programação: JavaScript, Python e JavaScript/HTML.

O funcionamento do WebSocket API para o Home Assistant contém estados do servidor:

1. Cliente estabelece ligação;
2. A fase de autenticação começa:
    a. Servidor envia uma mensagem de autenticação;
    b. Cliente envia mensagem;
    c. Caso a mensagem esteja correta passa para o passo 3;
    d. Caso a mensagem seja inválida passa para o passo 6;
3. Envia uma mensagem de confirmação;
4. Acaba a fase de autenticação;
5. Fase do comando começa;
    a. Cliente envia os comandos;
    b. Servidor responde;
6. Cliente ou servidor termina a sessão.

3.2.5. Entradas de configuração

As entradas de configuração [88] são os dados de configuração persistentemente armazenados no sistema do Home Assistant. A entrada de configuração é criada pelo utilizador através do UI (*User Interface*) do sistema, alimentado por um manipulador de fluxo de configuração, conforme definido pela componente. Também pode ter um manipulador de fluxos que fornece opções extras para a mesma componente.

Estas entradas de configuração constituem um ciclo de vida:

- Não carregado (*not loaded*), significa que a entrada de configuração não foi ainda carregada, isto é, entra na fase inicial quando a entrada de configuração é criada ou quando o sistema é reiniciado.

- Carregado (*loaded*), indica que a entrada de configuração é carregada.
- Erro de configuração (*error setup*), significa que ocorreu um erro ao tentar configurar a entrada de configuração.
- Nova tentativa de configuração (*error retry*), apresenta uma dependência da entrada de configuração que não estava, ainda, preparada. O sistema vai tentar de novo carregar no futuro. O tempo entre as tentativas poderão aumentar.
- Erro de migração (*error migration*), denota que a entrada de configuração foi migrada para uma nova versão e que a migração falhou.
- Falhou descarregamento (*failed unload*), afirma que a entrada de configuração tentou descarregar, no entanto, não suportou ou lançou uma exceção.

3.2.6. Entidades: integração de dispositivos e de serviços

Como foi mencionado anteriormente na secção 3.2.3, as integrações fazem parte dos dispositivos e dos serviços do Home Assistant. Os pontos de dados são representados como entidades que são padronizadas por outras integridades como luz, interruptor, entre outros. Estes vêm com serviços para controlo, mas a integração pode fornecer os seus próprios serviços caso não seja padronizado.

Uma entidade abstrai o funcionamento interno do Home Assistant, como integrador, não tem de se preocupar com os serviços ou estado da máquina. Em vez disso, estende a classe de uma entidade e implementa as propriedades e os métodos necessários para tipos de dispositivos que estejam integrados.

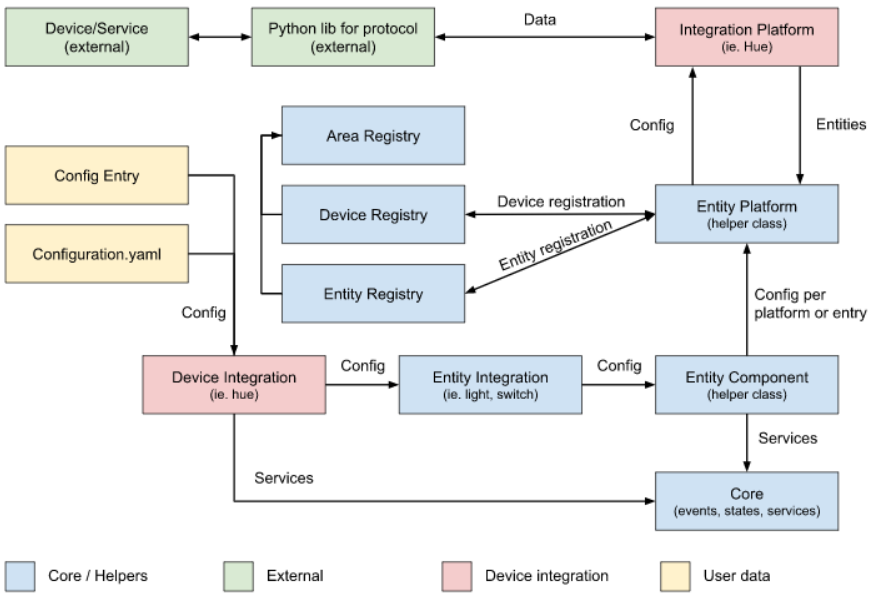

Figura 19. Sistema das entidades [88]

A configuração do dispositivo é fornecida pelo utilizador através das entradas de configuração ou recorrendo aos casos especiais através do ficheiro 'configuration.yaml'.

A integração do dispositivo, por exemplo Hue, como vemos na Figura 19, irá usar a configuração para estabelecer a ligação com o dispositivo ou serviço. Depois de encaminhar a entrada de configuração para configurar as suas respetivas integrações, que neste caso são a luz e o interruptor. Também é capaz de registar os seus serviços para partes que não estão padronizadas. Esses serviços são publicados sob domínio da integração, por exemplo na ativação do serviço.

A integração de uma entidade torna-se responsável por definir a classe de entidade abstrata e serviços para controlar as entidades. O ajudante do Componente de Entidade (*Entity Component*) fica responsável por distribuir a configuração das plataformas, encarrega-se pela descoberta e coleciona entidades para as chamadas de serviço.

O ajudante da Plataforma de Entidade (*Entity Plataform*) estabelece o controlo de todas as entidades da plataforma e pesquisa a existência de atualizações, se for necessário. Quando adiciona as entidades à plataforma de entidade, torna-se responsável por registar a entidade com o dispositivo e a entidade de registos.

A Plataforma de Integração usa configuração para pesquisa de um dispositivo ou de um serviço externo e cria entidades para serem adicionadas. Também é possível para as plataformas de integração registar os serviços de entidade. Estes serviços irão trabalhar em todas as entidades da integração do dispositivo para a integração de entidade, ou seja, para todas as entidades das luzes Hue. Os serviços são publicados sob o domínio da integração do dispositivo.

3.2.6.1. Interação de entidade com o Home Assistant Core

A classe da integração de entidade que herda da entidade base é responsável por buscar os dados e orientar os serviços de chamada. Se a votação estiver desativada, também fica responsável por indicar ao Home Assistant quando os dados ficam disponíveis.

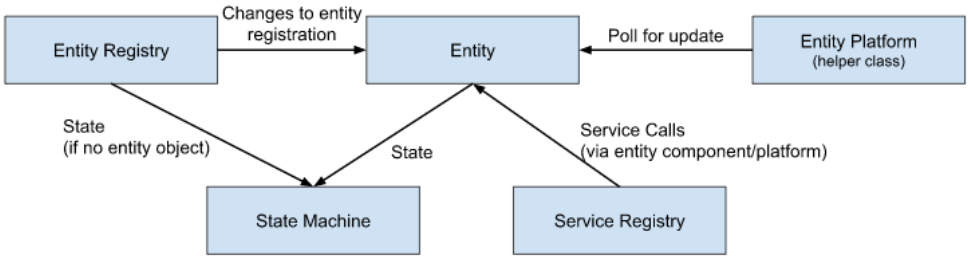

Figura 20. Interação entre entidades [88]

A base de entidade (Figura 20) é responsável por formatar os dados e escreve o estado da máquina. O registo de entidade indicará como estado indisponível para entidade registada que não é atualmente apoiado por um objeto de entidade.

3.2.6.2. Registo de entidade

O registo de entidade efetua o registo onde Home Assistant acompanha as entidades. Qualquer entidade que seja adicionada ao sistema que especifica o atributo "*unique_id*", será registada. Após o registo, existe a vantagem que a mesma entidade terá sempre a mesma identificação da entidade e as outras entidades nunca obterão a mesma identidade.

O utilizador tem a capacidade de sobrepor o nome duma entidade no registo de uma outra entidade. Quando prepara o nome no registo de entidade é usado a favor do nome do dispositivo.

3.2.6.2.1. Identificação Única

A identificação única nunca pode ser modificada pelos utilizadores nem pelos donos, porque o sistema poderá perder todas as definições que estejam relacionadas com as identificações únicas.

A entidade é pesquisada no registo com base numa combinação do tipo de plataforma e o nome da integração. As entidades não devem incluir o domínio e tipo de plataforma na identificação única como os sistemas, pois estes já consideram os identificadores.

Se o dispositivo tiver uma identificação única, mas fornece múltiplas entidades, acabam por combinar essa identificação com outros identificadores únicos para as entidades. Por exemplo, um dispositivo mede a temperatura e a humidade pode identificar exclusivamente as entidades usando '{*unique_id*}-{*sensor_type*}'.

Os requisitos aceitáveis para uma identificação exclusiva são:

- Número de série de um dispositivo;
- O endereço MAC;
- Latitude e longitude ou outra localização única;
- Identificador único.

3.2.6.3. Registo de entidade e desativando entidades

Existem opções no registo das entidades que impactam as entidades que interagem no núcleo, uma dessas opções é o "*disabled_by*". Com as integrações devem ser desativadas corretamente, enquanto estejam a ser executadas no sistema. Se as integrações mantêm as referências às entidades dos objetos criados, devem, então, registar essas referências apenas dentro do método do ciclo de vida da entidade "*async_added_to_hass*". Esse método de ciclo de vida apenas deve ser chamado, se a entidade estiver adicionada no Home Assistant.

A desativação de entidade funciona com as entidades fornecidas, através de uma entrada de configuração ou no ficheiro de configuração. Se a integração estiver configurada através de uma entrada de configuração e suporta descarregamento de entradas, o Home Assistant poderá recarregar a integração depois das entidades serem ativadas ou desativadas para aplicar as mudanças sem reiniciar o sistema.

Uma das formas para desativar as entidades é recorrer por parte do utilizador a editar no registo de entidade através da Interface do Utilizador. Isto só funciona para entidades que já estejam registadas.

Algumas integrações podem oferecer opções para o utilizador, de maneira a controlar quais as entidades que estão sendo adicionadas ao Home Assistant.

3.2.7. Registo do dispositivo

O registo do dispositivo é um registo que permanece o controlo dos dispositivos. Um dispositivo é representado no Home Assistant através de uma ou mais entidades. Por exemplo, um sensor que deteta temperatura e humidade expõe as entidades de temperatura, humidade e bateria.

3.2.7.1. O que é um dispositivo?

Um dispositivo no Home Assistant representa um dispositivo ou um serviço. Por exemplo, um termostato Ecobee com quatro sensores de sala equivale a cinco dispositivos no Home Assistant. Em cada dispositivo existe uma área específica geográfica e pode ter mais do que um input e um output.

Se ligar um sensor para outro dispositivo para ler os dados, deve ser representado como dois dispositivos diferentes. A razão para isso é que o sensor pode ser movido para ler os dados de outro dispositivo.

Um dispositivo que oferece múltiplos pontos finais e onde as partes do dispositivo que detetam ou emitem em áreas diferentes, devem ser divididos em dispositivos separados e referidos de volta ao dispositivo parente, através do atributo '*via_device*'. Isto permite que os pontos finais separados sejam atribuídos a diferentes áreas do edifício.

3.2.7.2. Definindo os dispositivos

Cada entidade é definida através da informação da propriedade. Esta propriedade é lida quando uma entidade é adicionada ao Home Assistant através da configuração. Um dispositivo será combinado com o dispositivo existente por meio de identificadores ou conexões fornecidas, tais como exemplo do endereço MAC ou número de séries. Se os identificadores e as conexões forem fornecidas, o registo do dispositivo tentará corresponder com os identificadores. Cada identificador e conexão são correspondidos individualmente.

```python
# Inside a platform
class HueLight(LightEntity):
    @property
    def device_info(self):
        return {
            "identifiers": {
                # Serial numbers are unique identifiers within a specific domain
                (hue.DOMAIN, self.unique_id)
            },
            "name": self.name,
            "manufacturer": self.light.manufacturername,
            "model": self.light.productname,
            "sw_version": self.light.swversion,
            "via_device": (hue.DOMAIN, self.api.bridgeid),
        }
```

Figura 21. Definindo o dispositivo [88]

Como se pode ver na Figura 21, os valores podem ser adicionados para o registo do dispositivo, se nenhum outro valor for definido. Isto pode ser usado pelas integrações que sabem alguma informação, mas não muito específica. Por exemplo, um router identifica o dispositivo pelo endereço MAC.

3.2.7.3. Removendo os dispositivos

As integrações podem permitir que o utilizador exclua um dispositivo na interface do utilizador. Para aplicar a remoção tem que ser implementado uma função assíncrona que configura essa remoção no módulo de Python, como podemos verificar na Figura 22.

```python
async def async_remove_config_entry_device(
    hass: HomeAssistant, config_entry: ConfigEntry, device_entry: DeviceEntry
) -> bool:
    """Remove a config entry from a device."""
```

Figura 22. Removendo o dispositivo [88]

Quando o utilizador elimina o dispositivo deve aguardar que aceita o pedido com True e a configuração de entrada será removido o dispositivo.

3.2.8. Home Assistant Core

Se quisermos implementar e desenvolver um ambiente virtual do Home Assistant Core, os *developers* recomendam alguns requisitos:

1. Instalar o Git;
2. Instalar o Docker;
    a. Para usar o Docker nos sistemas de Linux, macOS ou Windows 10 Pro/Enterprise/Education deve ser utilizado a versão atual do Docker;
    b. Caso esteja a utilizar Windows 10 Home, deve ser utilizado o WSL 2 e instalar também a versão mais recente do Docker Desktop. Estes passos podem ser aplicados também nos sistemas superiores do Windows 10 (Pro/Enterprise/Education);
3. Instalar o Visual Studio Code;
4. Instalar a extensão Remote – Containers.

Depois de ser aplicado *fork,* esta obtém a versão mais recente do Home Assistant Core, ou seja, é clonando o repositório no computador. Para os utilizadores que recorrem com Windows 10 devem colocar os seus ficheiros dentro do sistema dos ficheiros do WSL, e abrir o mesmo repositório com o Visual Studio Code.

De seguida, dentro da pasta “core”, devemos implementar um *script* para aplicar a instalação. Durante essa instalação, adiciona outros conteúdos de Python que devem ser utilizados dentro do Home Assistant Core.

```
josueferreira@ubuntu:~/core$ script/setup
Copy ./.vscode/settings.default.json to ./.vscode/settings.json.
Installing development dependencies...
Collecting wheel
  Downloading wheel-0.36.2-py2.py3-none-any.whl (35 kB)
Installing collected packages: wheel
Successfully installed wheel-0.36.2
Collecting awesomeversion==21.4.0
  Downloading awesomeversion-21.4.0-py3-none-any.whl (11 kB)
Collecting tox
  Downloading tox-3.23.1-py2.py3-none-any.whl (85 kB)
     |████████████████████████████████| 85 kB 2.6 MB/s
Collecting tox-pip-version
  Downloading tox-pip-version-0.0.7.tar.gz (8.4 kB)
Collecting colorlog
  Downloading colorlog-5.0.1-py2.py3-none-any.whl (10 kB)
Collecting pre-commit
  Downloading pre_commit-2.13.0-py2.py3-none-any.whl (190 kB)
     |████████████████████████████████| 190 kB 3.7 MB/s
Collecting mypy==0.812
  Downloading mypy-0.812-cp38-cp38-manylinux2010_x86_64.whl (22.5 MB)
     |████████████████████████████████| 22.5 MB 12.6 MB/s
Collecting stdlib-list==0.7.0
  Downloading stdlib_list-0.7.0-py3-none-any.whl (60 kB)
     |████████████████████████████████| 60 kB 9.3 MB/s
Collecting tqdm==4.49.0
  Downloading tqdm-4.49.0-py2.py3-none-any.whl (69 kB)
     |████████████████████████████████| 69 kB 10.2 MB/s
Collecting pipdeptree==1.0.0
  Downloading pipdeptree-1.0.0-py3-none-any.whl (12 kB)
Collecting six>=1.14.0
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Collecting virtualenv!=20.0.0,!=20.0.1,!=20.0.2,!=20.0.3,!=20.0.4,!=20.0.5,!=20.0.6,!=20.0.7,>=16.0.
0
```

Figura 23. Instalação das extensões de Python

Para experimentar o sistema *core*, recorremos numa máquina virtual (Ubuntu 20.04.2.0 LTS) e seguimos as mesmas instruções e os comandos para finalizar a configuração.

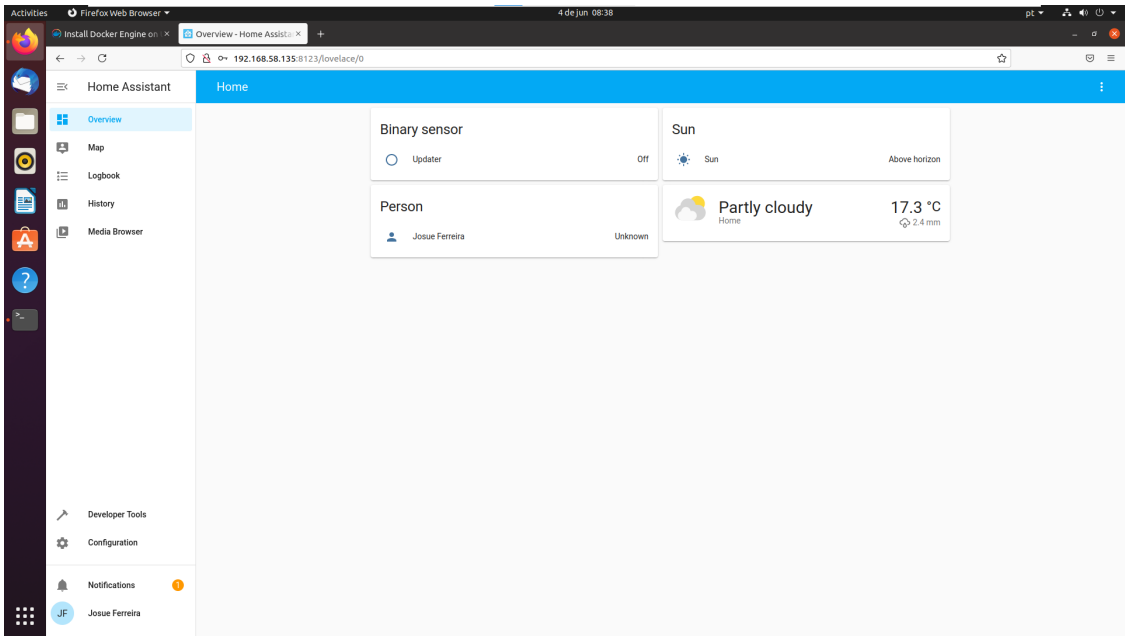

Figura 24. Página principal do Home Assistant Core

A instalação acontece da mesma forma nas outras versões do Home Assistant e verifica a existência de dispositivos inteligentes, sensores e outros semelhantes. No entanto, nesta versão, a interface não apresenta a possibilidade de instalar extensões ou de fazer atualizações.

Cada vez que queremos utilizar o Home Assistant Core, teremos que utilizar comandos para executar o sistema.

3.2.8.1. Submissão do trabalho

Os *developers* recomendam que devemos utilizar o Git e ter uma conta no GitHub, caso queiramos obter as últimas atualizações ou contribuir para o projeto do Home Assistant Core.

Na criação de um novo *branch* aplicamos novas mudanças, de maneira a criar uma nova plataforma, desenvolver uma nova integração, corrigir falhas e testar as mudanças. Após a verificação do sistema, deve ser aplicado o *push* à nova *branch.* Para finalizar, navegamos pelo repositório principal e escolhemos os *branches* que fazem parte das nossas modificações.

3.2.8.2. Criando as primeiras integrações

O Home Assistant tem integrações que estão embebidas no sistema. Após o arranque do sistema, estas podem ser descobertas e depois, por opção, podem ser adicionadas. Caso essas integrações não sejam descobertas, teremos que adicioná-las manualmente.

```
python3 -m script.scaffold integration
```

Figura 25. Comando para permitir integrações [88]

O comando que podemos visualizar na Figura 25 cria e configura uma integração que pode estar apta para a interface. Podemos, ainda, utilizar vários exemplos de *scripts* de Python que os *developers* fornecem no GitHub.

Para entender, inicialmente, a funcionalidade do sistema com as suas integrações, os *developers* forneceram *scripts* básicos como *"Hello World"*, de forma síncrona e assíncrona, como se pode observar na Figura 26 e na Figura 27.

```
DOMAIN = "hello_state"


def setup(hass, config):
    hass.states.set("hello_state.world", "Paulus")

    # Return boolean to indicate that initialization was successful.
    return True
```

Figura 26. "Hello World" de forma síncrona [88]

```python
DOMAIN = "hello_state"


async def async_setup(hass, config):
    hass.states.async_set("hello_state.world", "Paulus")

    # Return boolean to indicate that initialization was successful.
    return True
```

Figura 27. "Hello World" de forma assíncrona [88]

O Home Assistant procura as integrações em duas localizações específicas:

- "config/custom_components/<domínio>";
- "homeassistant/componentes/<domínio>" (para integrações embebidas).

Também é possível sobrepor a localização do domínio das integrações "config/custom_components" para outra localização.

Geralmente, dentro da pasta, quando implementamos uma integração, existem dois ficheiros para configurar o mesmo, estas são: o ficheiro JSON ("manifest.json") e o modulo de Python ("__init__py"). Em cada ficheiro, que contenha integrações, tem o ficheiro JSON pois este representa informação básica sobre a integração.

```json
{
  "domain": "hue",
  "name": "Philips Hue",
  "documentation": "https://www.home-assistant.io/components/hue",
  "issue_tracker": "https://github.com/balloob/hue/issues",
  "dependencies": ["mqtt"],
  "after_dependencies": ["http"],
  "codeowners": ["@balloob"],
  "requirements": ["aiohue==1.9.1"],
  "quality_scale": "platinum",
  "iot_class": "local_polling"
}
```

Figura 28. Exemplo de ficheiro manifest [88]

Dentro desse ficheiro, tal como podemos analisar na Figura 28, deve conter informações essenciais que fazem com que a integração funcione.

As informações do *manifest* são os seguintes:

- Domínio, que consiste num nome curto com carateres e linhas sublinhadas. Este domínio deve ser considerado único e não pode ser modificado;

- Nome, que refere o nome da integração;
- Versão, as versões para as integrações principais devem ser implementadas;
- Documentação, o *website* contém documentação de como deve utilizar a integração;
- Rastreador de problemas deve ser utilizado quando os utilizadores detetam uma falha. Se a integração foi submetida no Home Assistant, esta deve ser, também, omitida e caso sejam integrações embutidas estas são geradas automaticamente;
- Dependências, são outras integrações do Home Assistant que devem ser configuradas com sucesso antes da integração ser carregada. Isto, pode ser considerado necessário quando queremos oferecer funcionalidades para outras integrações que recorram com os *webhooks* e as ligações MQTT;
- Pós-dependências, esta opção é usada especificamente para dependências que podem ser usadas para a integração, mas não são essenciais;
- Proprietários de código identificam os utilizadores do GitHub ou nomes das equipas que foram responsáveis pela integração;
- Fluxo de configuração especifica a chave do fluxo de configuração, se a integração tem a configuração para criar uma entrada de configuração. Quando é especificado o ficheiro "config_flow.py" deve existir para a integração.

O módulo de Python, ("__init__.py") é um componente de ficheiro. Se a integração apenas fornece uma plataforma, então pode ficar limitado introduzindo a integração.

Existem outros ficheiros que podem ser implementados para as integrações, que são dispositivos e serviços.

Se integrar um ou mais dispositivos, tem de ser criado uma plataforma que consiga interagir com uma integração de entidade. Essas entidades podem ser: luzes, comutadores, dispositivos climáticos, entre outros. Para essa plataforma deve ser criado um ficheiro com domínio da integração. Na construção de uma integração, e se considerarmos a integração como "*light*", por exemplo, devemos criar um ficheiro de Python com o nome de "light.py" dentro do mesmo ficheiro.

Se precisarmos de registar os serviços, devemos fornecer uma descrição dos serviços disponíveis. A descrição fica armazenada no ficheiro YAML dos serviços ('*services*'). Esse ficheiro contém serviços prontos para várias situações. Os serviços podem ser chamados através das automações e para o serviço "*Developer tools*" do Home Assistant.

```
# configuration.yaml entry
hello_service:
```

Figura 29. Configuração para o serviço [88]

Para carregar a integração tem de ser adicionado no ficheiro YAML da configuração e quando o componente for carregado deve ficar disponível para chamar.

Os serviços são considerados úteis, se o utilizador tiver conhecimento dos mesmos. O Home Assistant usa o ficheiro de serviços como parte da integrante na descrição dos serviços. Estes são publicados sob o domínio do nome da integração, portanto apenas pode ser usado o nome do serviço como chave base.

3.2.9. Home Assistant Supervisor

O Supervisor [87, 88] permite o utilizador controlar a instalação do Home Assistant. Este tem as seguintes responsabilidades:

- Executar o Home Assistant Core;
- Atualizar o Home Assistant Core e se falhar, automaticamente regressa à versão anterior;
- Faz com que recupere *backups*;
- Aplica um sistema de áudio unificado;
- Atualiza o sistema operativo do Home Assistant (desativado na versão Supervised).

3.2.9.1. Arquitetura

Na Figura 28, este sistema operativo tem plataformas que representam funcionalidades essenciais que fazem com que execute o mesmo.

- Home Assistant Core, a plataforma de automação residencial;
- *Add-nos* ou Extensões, são aplicações extras que o utilizador deseja que execute no servidor;
- DNS, permite o núcleo e as extensões para comunicar entre um e outro;
- Áudio, permite o núcleo e as extensões para tocar áudio;
- DNS, ajuda a descobrir e estabelece ligações para dispositivos e serviços na rede;
- Supervisor controla todas as partes do sistema e mantém tudo atualizado;
- Docker é o serviço contentor para executar as aplicações;
- Sistema operativo recorre a um sistema base Linux;

- D-Bus é um sistema de comunicação para controlar partes de sistemas operativos, tais como gestão de redes.

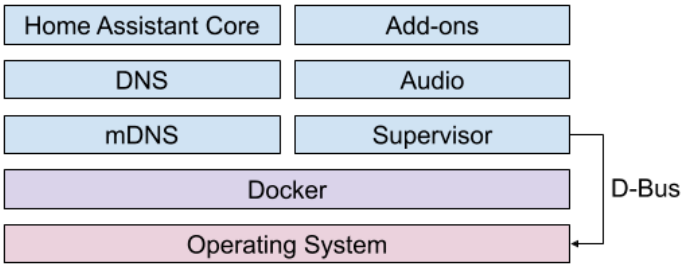

Figura 30. Arquitetura do Supervisor [88]

3.2.10. Home Assistant Supervised

Esta versão fica implementada no sistema Linux, base Debian. Temos acesso completo, como na versão Home Assistant OS. Contudo, a versão Supervised serve para utilizadores que tenham conhecimento avançado nos sistemas operativos Linux/Unix, no Docker e na gestão de redes.

3.2.10.1. Instalação do sistema

Para começar, adicionamos uma máquina virtual ou um *bare metal*, na mesma versão do sistema operativo Linux que foi instalado o Home Assistant Core, como foi mencionado na secção 3.2.8. Após a instalação, deve ser utilizado os comandos essenciais no terminal para atualizar o sistema.

No passo seguinte deve ser instalado e testado o Docker. Antes de aplicar a instalação o Home Assistant Supervised deve ser instalado algumas dependências no sistema Linux, como os *developers* indicam na página do GitHub. Finalmente, desativa-se o ModemManager e usamos o *script* para instalar o Home Assistant Supervised [89].

Após a instalação do Home Assistant Supervised, o terminal do sistema operativo indica informações para aceder ao sistema através do browser.

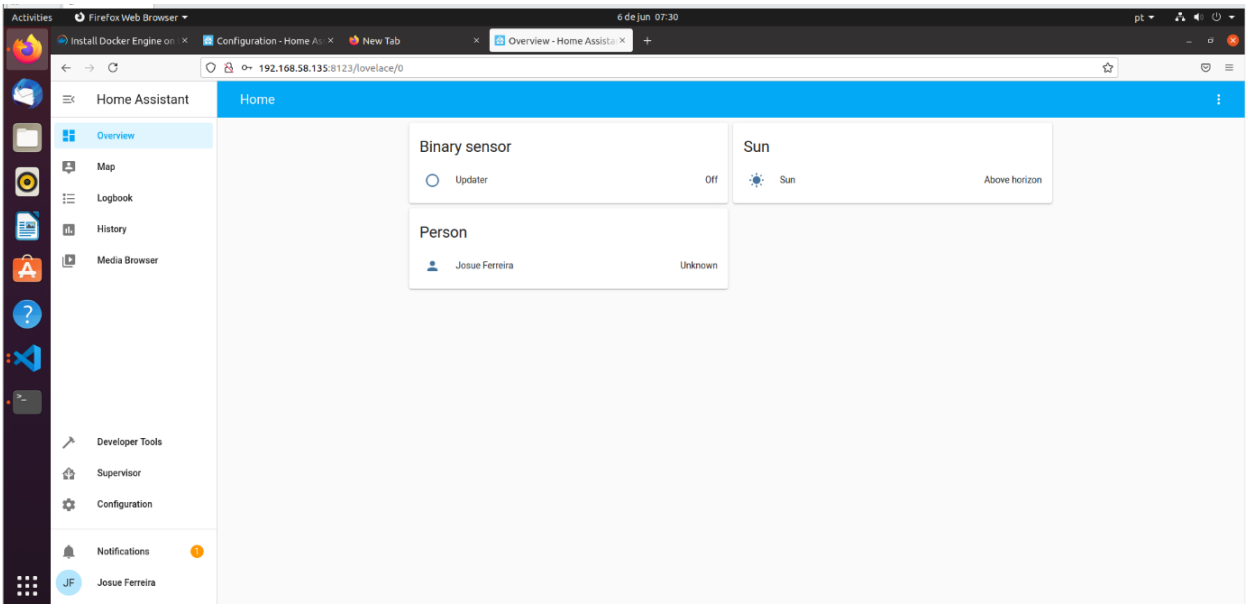

Figura 31. Hub do Home Assistant Supervised

Este sistema tem algumas vantagens ao compararmos com o sistema anterior (Home Assistant Core), tais como: não exige nenhum método para ativar constantemente o ambiente virtual, através de comandos no terminal e temos a possibilidade de adicionar extensões. Portanto, tem quase todas as funcionalidades do Home Assistant OS.

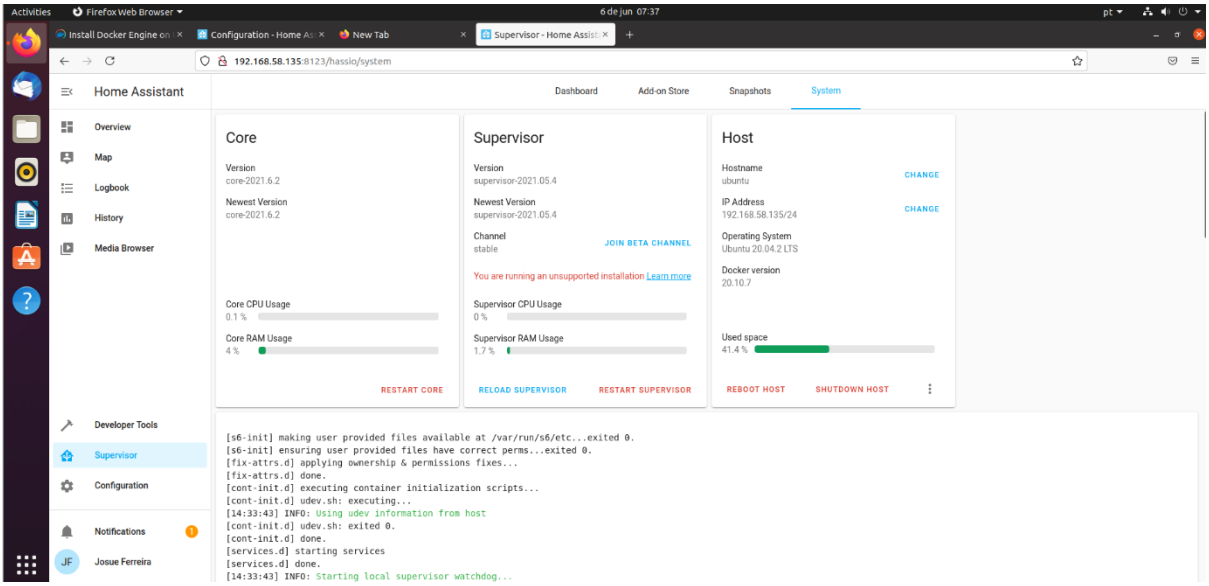

Figura 32. Indicação da versão Supervised

Contudo, este sistema tem uma limitação, a gestão do sistema está completamente dependente do conhecimento do utilizador. Caso o sistema Linux ou o sistema Home Assistant tenham alguma falha, têm que ser corrigidos manualmente.

A Figura 32 indica que estamos a executar uma instalação que "não é suportada", o que prova que este sistema é a versão Supervised.

Outra parte essencial que o utilizador deve prestar atenção é a atualização do Home Assistant Supervised, pois, ao contrário do Home Assistant OS, não se consegue atualizar facilmente. Por outras palavras, quando o sistema notifica a existência de alguma atualização, o utilizador tem que atualizar primeiro o sistema operativo, o sistema Linux. Se não aplicar a atualização do sistema Linux, o Home Assistant Core indica a impossibilidade de atualizar devido ao estado "não saudável", como é visível na Figura 33.

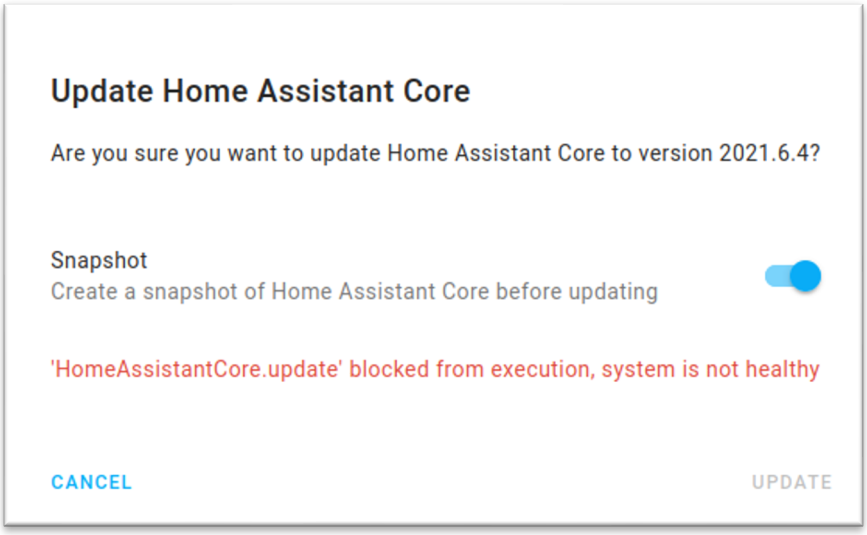

Figura 33. Prevenção de atualização do sistema

A mesma parte acontece quando tentamos atualizar com as extensões do sistema, devido ao motivo que foi mencionado na figura anterior. Conseguimos observar o mesmo na Figura 34, para a extensão Samba Share.

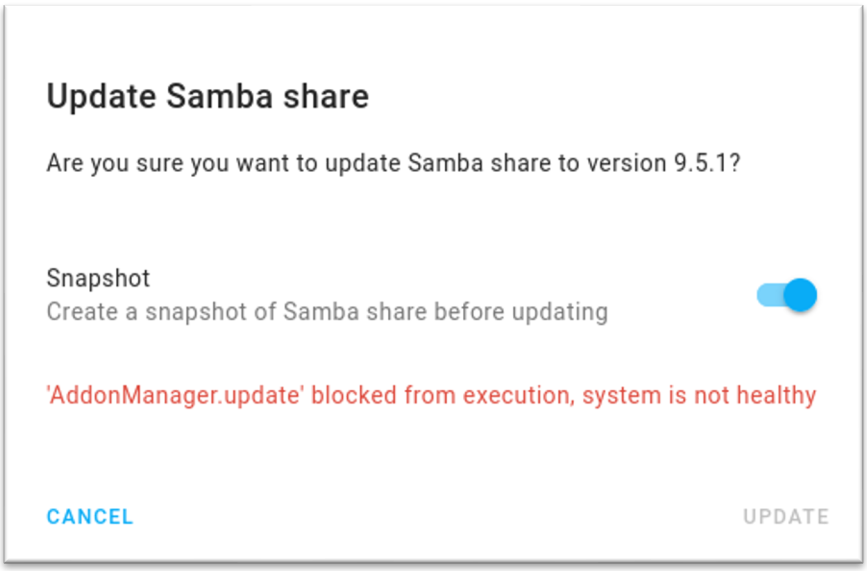

Figura 34. Prevenção de atualização da extensão

3.2.10.2. *Add-ons*

Para criar uma extensão dentro do Home Assistant, quer na versão Supervised quer na versão OS, existem duas formas:

- Samba Share, facilita o acesso dos ficheiros do Home Assistant quando utilizamos os sistemas Windows e macOS;
- SSH Terminal, em que se acede ao sistema através do terminal, usando um túnel SSH.

Ao usarmos a extensão Samba Share teremos primeiro que configurar as seguintes partes:

- Criar o nome do Workgroup ou manter o nome pré-definido "WORKGROUP";
- Adicionar o nome do utilizador e a palavra-chave para aceder ao sistema. Opcionalmente, podemos manter a palavra-chave como vazio ou "*null*" no ficheiro da configuração da extensão;
- Antes de ativar, devemos verificar o endereço de IP do sistema Windows e/ou macOS para adicionarmos esse IP na matriz "*allow_hosts*".

Após a instalação do Home Assistant, não é possível encontrar a extensão SSH Terminal, pois o modo avançado fica desativado de forma pré-definida. Após a ativação deste modo, podemos aplicar a instalação da extensão e configuramos a palavra-chave para ativar o mesmo.

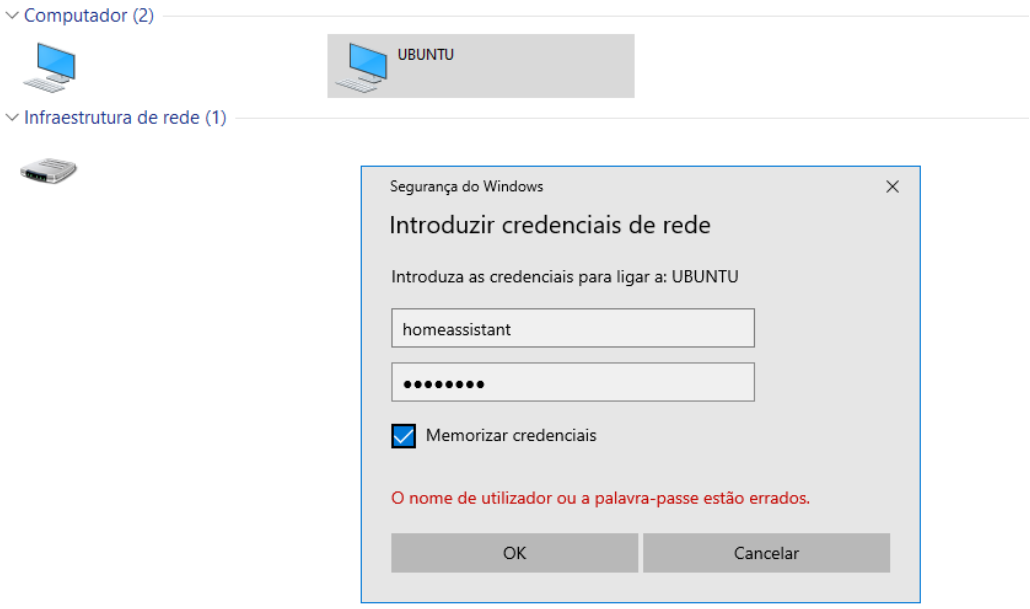

Figura 35. Acesso ao Home Assistant com Samba Share

Começando com a criação da extensão, tivemos que fazer uma pasta e geramos três ficheiros: um ficheiro de configuração JSON ("config.json"), um ficheiro Docker ("Dockerfile") e um ficheiro *bash* ("run.sh").

```
ARG BUILD_FROM
FROM $BUILD_FROM

ENV LANG C.UTF-8

# Copy data for add-on
COPY run.sh /
RUN chmod a+x /run.sh

CMD [ "/run.sh" ]
```

Figura 36. Ficheiro Docker [88]

```
{
 "name": "Hello world",
 "version": "1",
 "slug": "hello_world",
 "description": "My first real add-on!",
 "arch": ["armhf", "armv7", "aarch64", "amd64", "i386"],
 "startup": "application",
 "boot": "auto",
 "options": {},
 "schema": {}
}
```

Figura 37. Ficheiro de configuração [88]

```
#!/usr/bin/with-contenv bashio

echo Hello world!
```

Figura 38. Ficheiro bash script [88]

As figuras 36, 37 e 38 contêm as funcionalidades básicas para executar um "*Hello World"* numa extensão do Home Assistant Supervised. Recorrendo à mesma extensão, são adicionadas outras funcionalidades para a extensão que irá implementar um servidor.

```
# Install requirements for add-on
RUN apk add --no-cache python3

# Python 3 HTTP Server serves the current working dir
# So let's set it to our add-on persistent data directory.
WORKDIR /data
```

Figura 39. Modificação do ficheiro Docker [88]

Revendo o ficheiro Docker, na Figura 39, adiciona-se os requisitos para execução do servidor HTTP com Python 3. No ficheiro de JSON (Figura 40) são adicionados os esquemas essenciais para configurar as portas ('*ports'*), as opções ('*options*') e o esquema ('*schema*'). Finalmente, é configurado uma *script* de Python no ficheiro *bash*.

```
"ports": {
  "8000/tcp": 8000
}
"options": {
  "beer": true,
  "wine": true,
  "liquor": false,
  "name": "world",
  "year": 2017
},
"schema": {
  "beer": "bool",
  "wine": "bool",
  "liquor": "bool",
  "name": "str",
  "year": "int"
},
```

Figura 40. Modificações para o ficheiro JSON [88]

Após essas modificações, alteramos o ficheiro *bash* que utilizamos na extensão básica para finalizar a modificação da extensão. Chamamos o comando do Python 3 para executar o servidor HTTP e, finalmente, conseguimos obter resposta por parte do servidor usando o endereço http://homeassistant.local:8000 ou usando o endereço IPv4 com a porta 8000.

```
python3 -m http.server 8000
```

Figura 41. Comando para executar o servidor no ficheiro bash [88]

Após a restante configuração, podemos transferir a extensão customizada do Home Assistant com o Samba Share. De seguida, é adicionada a extensão do *"Hello World"*.

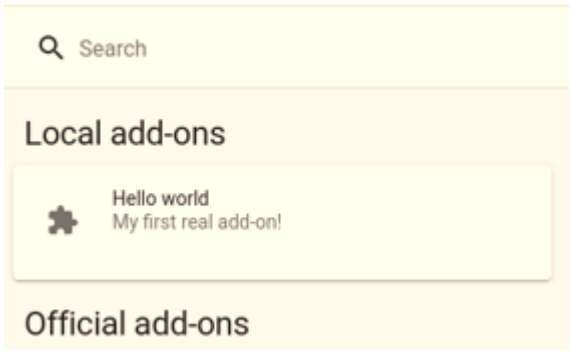

Figura 42. Extensão local do HA, parte 1

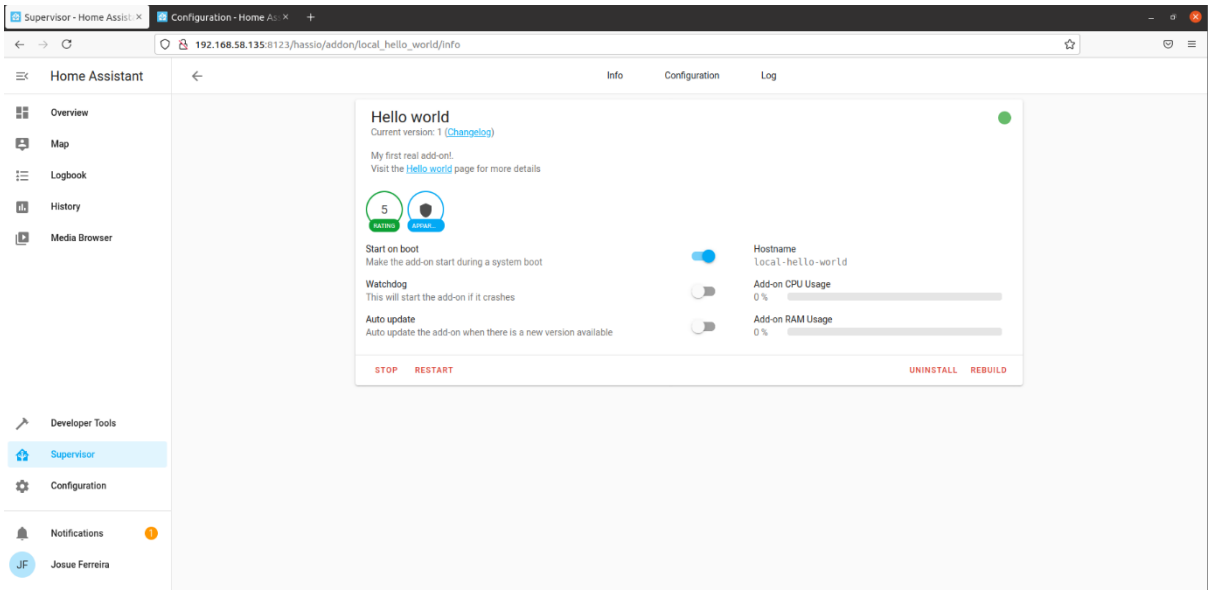

Figura 43. Extensão local do HA parte 2

Depois da ativação da extensão pode-se chamar pelo browser a porta 8000, como podemos ver a Figura 43.

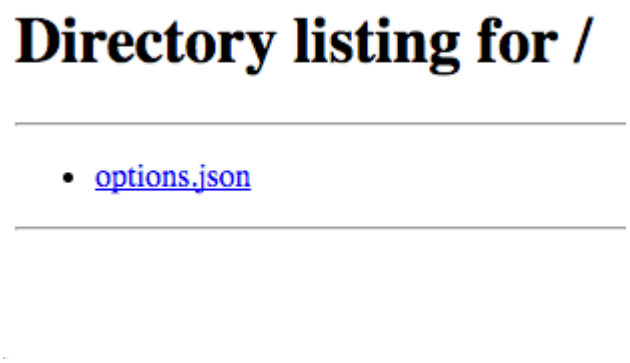

Figura 44. Servidor ativado

Na extensão pode-se modificar as funcionalidades do esquema usando o YAML para mudar o formato, contudo, ao adicionar novo conteúdo poderá não ter efeito, como se pode ver na Figura 45.

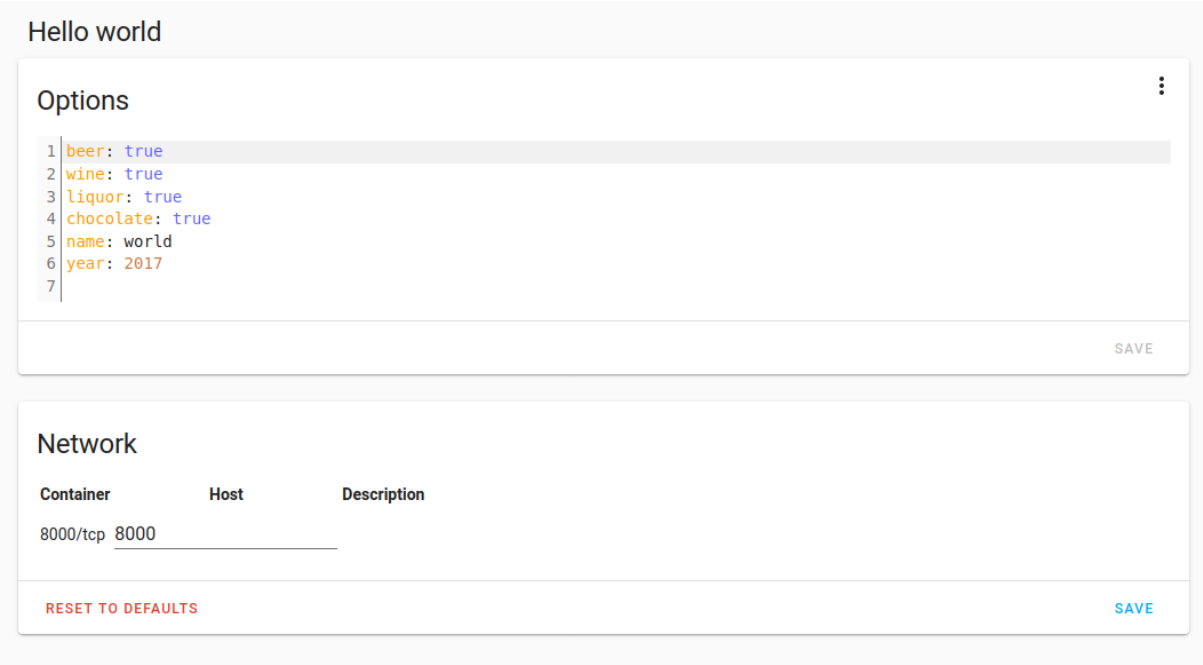

Figura 45. Configuração da extensão

Para implementar outras informações, por exemplo no ficheiro JSON, teria que ser desinstalada a extensão anterior.

3.2.11. Home Assistant Operating System (HAOS)

Esta versão do Home Assistant Operating System (HAOS) [88] é um sistema operativo desenvolvido especificamente para executar o Home Assistant nos computadores *single board* e nos sistemas x86-64. Tem como objetivo fornecer um sistema operativo robusto e livre de manutenção para executar o Home Assistant.

O HAOS não utiliza uma distribuição clássica Linux, em vez disso, foi implementado um sistema Buildroot [91], que fornece uma infraestrutura sistema Linux. Contudo, Buildroot permite uma compilação cruzada para arquiteturas diferentes. Isto torna-se útil para compilar arquiteturas que tenham poucos recursos, como por exemplo as arquiteturas ARM (*Acorn RISC Machine*). Este sistema também contém ferramentas regulares de Linux e *software* GNU.

3.2.11.1. Instalação

Na parte da instalação deste sistema podemos aplicar duas formas:

- Instalar o sistema do Home Assistant ao formatar num cartão SD, ou instalar numa máquina virtual ou num sistema x86-64;
- Instalar manualmente através do repositório do GitHub, mas como pré-requisito terá que instalar o Docker container para fazer o resto da instalação.

3.2.11.2. Partição

No HAOS existe um esquema de partição diferente que é tipicamente usado no sistema Linux.

O HAOS tem uma preferência de usar GPT (GUI *Partition Table*). As execuções ROM para alguns SoCs não suportam GPT. Neste caso, é usada uma tabela de partição GPT/MBR. Caso contrário, é usado a *legacy* MBR.

3.2.11.2.1. Partição dos sistemas

A partição de arranque tipicamente é partição FAT e contém conteúdo específico do sistema para ativar o arranque. Para os sistemas x86-64 é o sistema de partição EFI que tem Barebox.

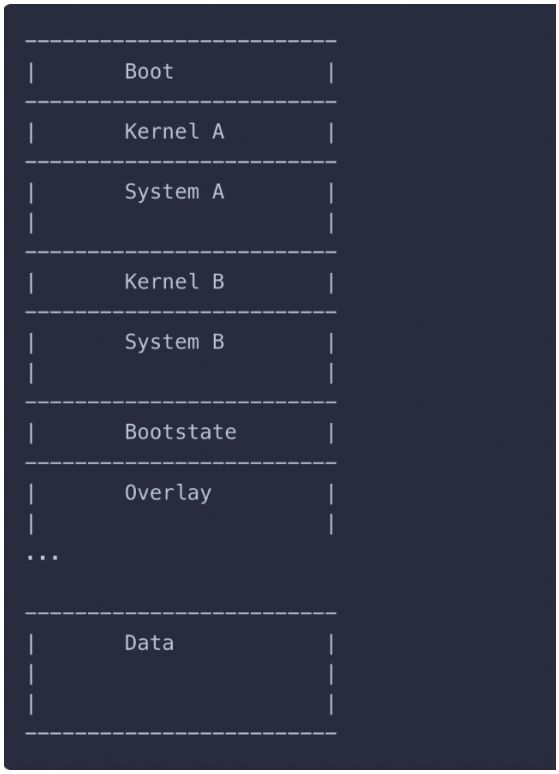

Figura 46.Exemplo da partição [88]

As próximas duas versões: o *kernel* do Linux e os sistemas operativos principais, estão armazenados (Kernel A/B e Sistema A/B, sendo no total 4 partições), como podemos ver na Figura 46. Isto permite que o sistema volte à versão anterior, caso o novo sistema operativo falhe, usando o método *update* A/B [92]. As partições do sistema apenas são escritas durante a atualização e só realizam a leitura nas operações regulares.

3.2.11.2.2. Partição de dados

A partição de dados é a partição principal que contém todos os containers (Supervisor/Core/Plug-ins e *Add-nos*) e os dados do utilizador. A maioria das operações de entrada e saída (I/O) beneficiam mais se for feito um armazenado rápido. A montagem no "/mnt/data" e algumas subdiretorias são montadas noutros sítios (por exemplo "/var/lib/docker"). O sistema de ficheiros "hassos-data" é usado para encontrar e montar a partição [88].

Após uma instalação, os dados de partição contêm as últimas versões do Supervisor e os Plugs-ins, não contêm o Home Assistant Core pré-instalado. O Supervisor faz o download da versão mais recente do Home Assistant Core [88].

O recurso do disco de dados usa o "hassos-data". O recurso prepara o disco para dividir e criar um sistema de ficheiros "hassos-data-external". Ao reiniciar a utilidade do sistema de ficheiros "dumpe2fs" é usado para transferir todos os dados existentes da partição "hassos-data" anterior para a nova partição.

Finalmente, o sistema de ficheiros com dados existentes passa para "hassos-data-old", de maneira a não executar novamente, e a nova partição passa para "hassos-data".

3.3. Hubitat

O sistema Hubitat tem uma documentação [92] que começa da instalação básica do aparelho até às documentações para *developers*. Isto é uma documentação simples com imagens incluídas que são aptas para qualquer utilizador.

A documentação apresenta conteúdo essencial para começarmos a usar o Hubitat. Resumidamente, separamos o conteúdo nestas seguintes partes:

- Instalação e configuração do sistema;
- Descobrindo os dispositivos;
- Serviços de subscrição;

- Código para os drivers;
- Documentação das aplicações;
- Documentação dos dispositivos;
- Guias '*How-to*';
- Documentação de *developers.*

3.3.1. Instalação e configuração do sistema

A instalação apresenta os primeiros passos para configurar o aparelho, navegar e explorar a existência de dispositivos que existem na rede de casa. Percebemos logo que tem instruções explícitas para explorar os mecanismos, quando mexemos pela primeira vez o aparelho.

3.3.2. Descobrindo os dispositivos

Para explorar os dispositivos temos opções para os encontrar, como podemos ver na Figura 47, onde podemos incluir ou excluir os aparelhos.

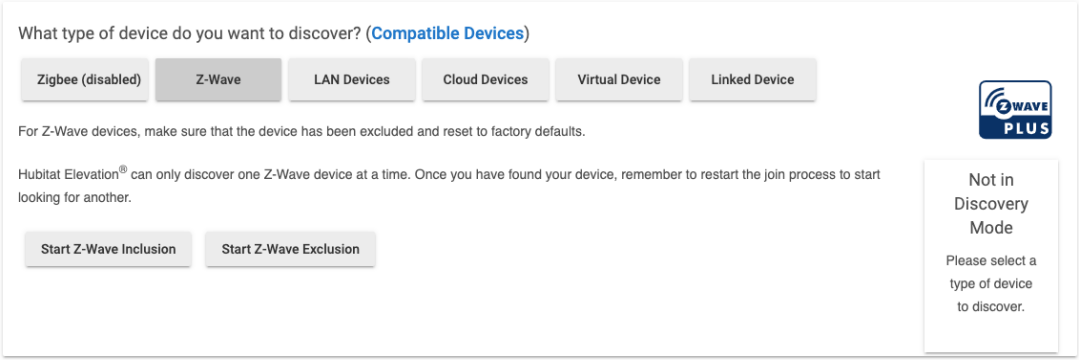

Figura 47. Modo de inclusão e exclusão de Z-Wave [92]

Para implementar os dispositivos de casas inteligentes em aplicações como Amazon, Ecobee e outros controladores, basta instalar a aplicação do Hubitat, como podemos analisar na Figura 48.

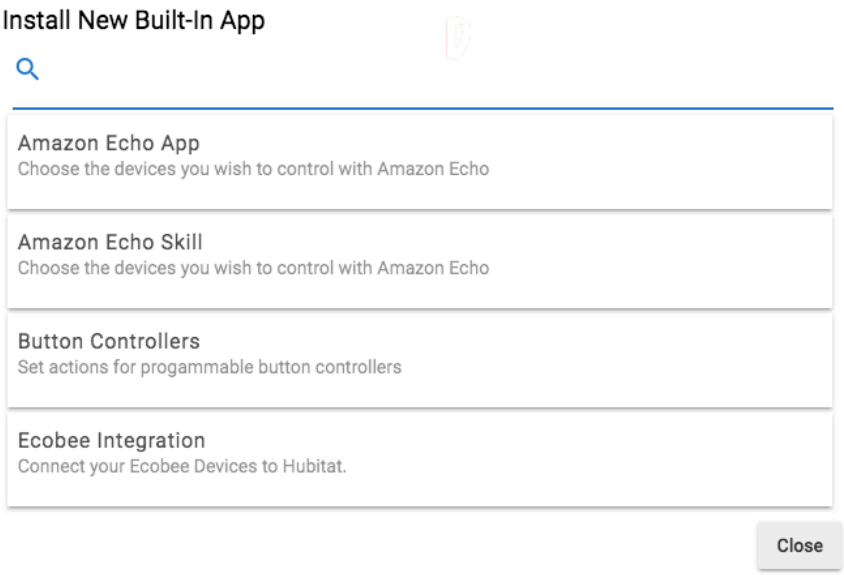

Figura 48. Instalação de uma aplicação [92]

3.3.3. Subscrição de Serviços

Na seção da Subscrição de Serviços, apresenta-se a opção de os utilizadores interagirem com os dispositivos remotamente. Portanto, o Hubitat tem uma subscrição para preservar as regras e outras funcionalidades no sistema.

Na parte da subscrição existem duas opções:

- Hub Protect;
- Remote Admin.

O Hub Protect utiliza uma subscrição anual que alcança no máximo cinco backups de *cloud*, incluindo as regras de automação, as definições e os dispositivos. Também podemos migrar para outro Hub, evitando que configuremos, novamente, as regras e os dispositivos.

O Remote Admin utiliza uma subscrição mensal que monitoriza todos os *hubs*. Isto facilita a gestão remota, capaz de configurar ou ajustar os dispositivos e as regras enquanto estiver fora de casa.

3.3.4. Código das aplicações

Os *developers* inserem as aplicações, normalmente, no GitHub e tem que ser copiado o código para implementar no sistema do Hubitat.

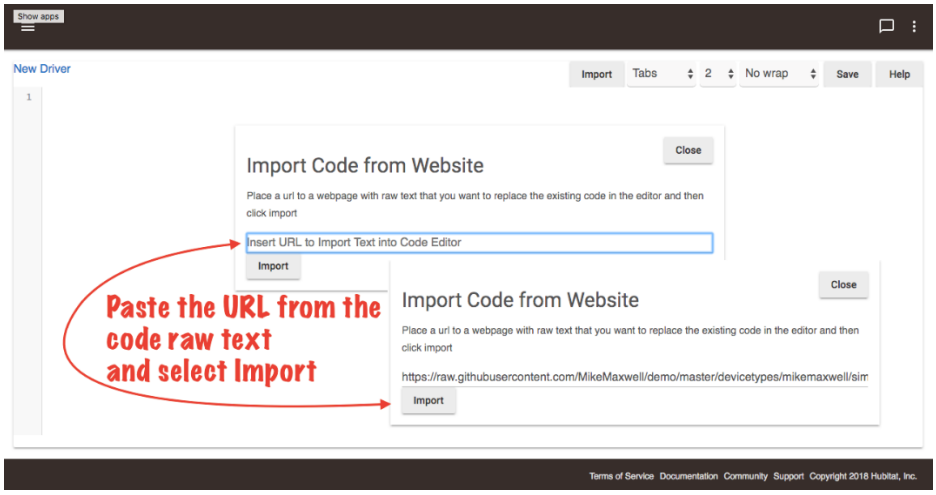

Figura 49. Importar o código [92]

Podemos inserir o código fonte ou, em alternativa, adicionamos o URL, como podemos verificar na Figura 49. Por vezes, pode haver algum erro por *default* do sistema, resultando com que não guarde as implementações da nova aplicação.

Algumas aplicações precisam de acesso OAuth (*Open Autorization*) e têm que ser ativados a opção OAuth para gerar a chave. Após a adição da aplicação teremos que procurar a aplicação feita pelo utilizador e passa a executar a aplicação, como podemos observar nas Figuras 50 e 51.

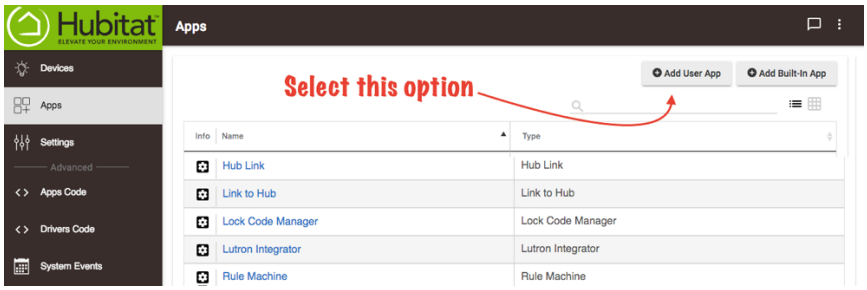

Figura 50. Adicionar uma aplicação feita pelo utilizador [92]

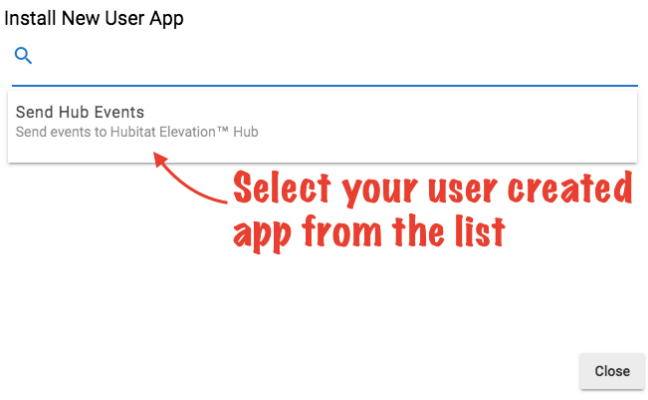

Figura 51. Procurar a aplicação feita pelo utilizador [92]

3.3.5. Código de *drivers*

Na secção "*Drivers Code*" recorremos aos mesmos métodos de configuração das aplicações. Ao selecionarmos no "*New Driver*", como observamos na Figura 52, podemos importar o código do fabricante ou copiar o URL do GitHub, como acontece na Figura 49.

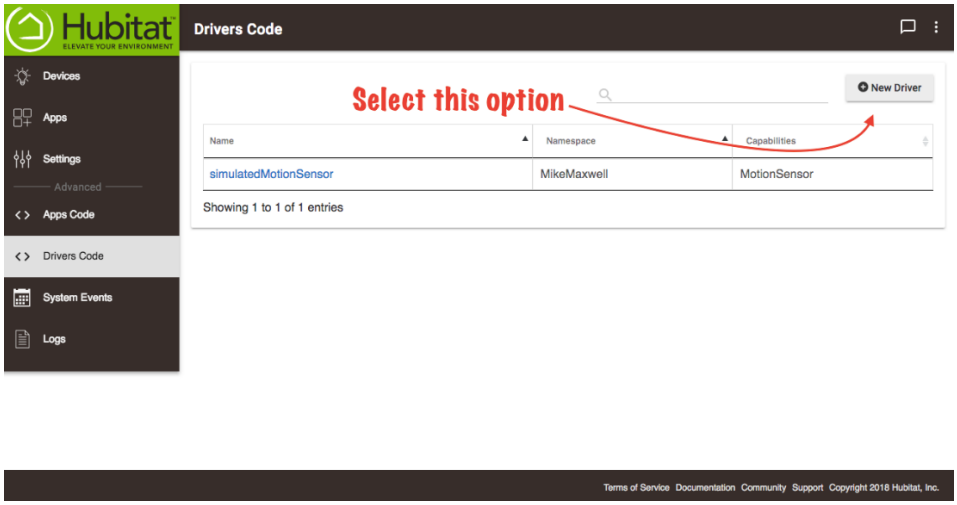

Figura 52. Adicionar novas drivers [92]

Depois de ser implementado com sucesso os drivers, teremos que adicionar um novo dispositivo que foi acrescentado pelo utilizador, como vemos nas Figuras 53 e 54.

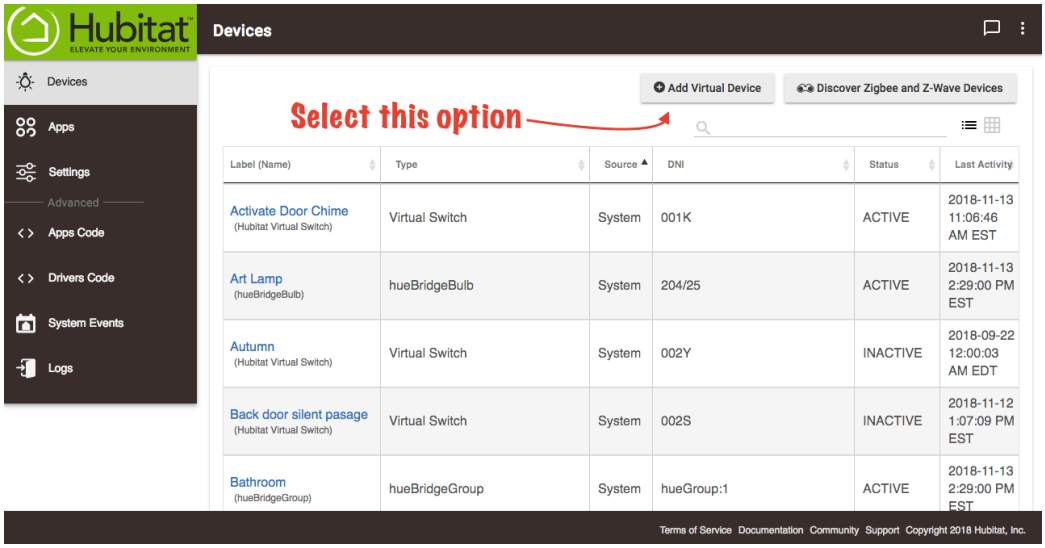

Figura 53. Adicionar novo driver [92]

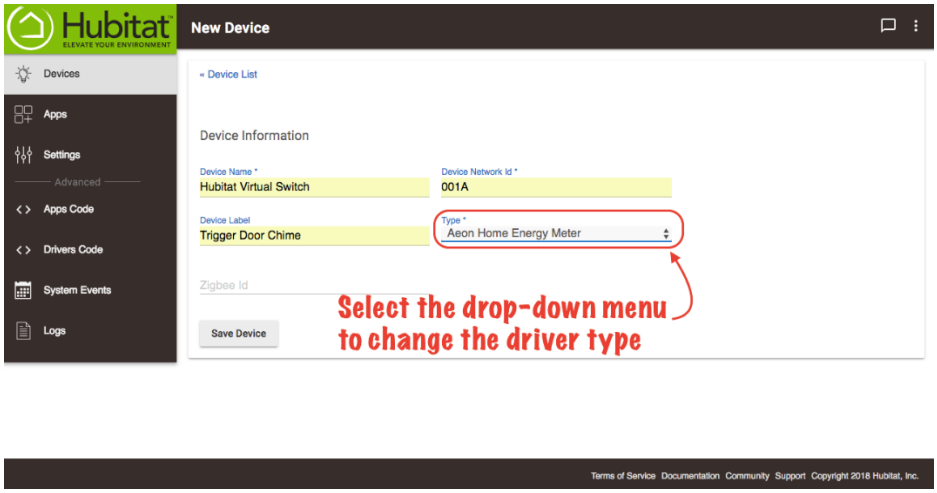

Figura 54. Procurar novo driver [92]

3.3.6. Documentação de dispositivos e de aplicações

Tanto o Hubitat como o Home Assistant têm uma documentação que explicam como devemos configurar as aplicações e os dispositivos. Por exemplo, como foram utilizados o Philips Hue Bridge e as luzes Philips Hue, recorremos a documentação do "*Philips Hue integration*" [93]. As instruções para configurar a aplicação são facultativas e explícitas para qualquer utilizador.

3.3.7. Guias "*How to*"

Nesta documentação, também podemos experimentar outras funcionalidades do sistema, como por exemplo a configuração de uma rede *Mesh* com os protocolos Zigbee e Z-Wave, a realização de um *backup* ou atualização do sistema e implementação de notificações.

3.3.8. Documentação dos *developers*

Nesta documentação apresentamos a estrutura, a interfaces da aplicação, as interfaces dos drivers e a estrutura das bibliotecas do sistema. Existem, ainda, métodos e classes para aprofundar o estudo das aplicações e os protocolos do Hubitat.

Conforme se explora na documentação dos *developers* e nos fóruns do Hubitat, os *developers* explicam que o sistema recorre a linguagem de programação Apache Groovy que usa uma sintaxe semelhante à da linguagem de programação Java.

Os utilizadores que já tiveram experiência de programação no sistema Samsung SmartThings [93] afirmam que a documentação desse sistema é um bom começo para desenvolver programação no Hubitat. Também, existem alguns exemplos através do GitHub

[94] para desenvolver aplicações e drivers customizados no Hubitat. Mesmo que a companhia forneça exemplos básicos, as aplicações e os drivers nativos não são completamente publicados [95] e, por ser uma pequena companhia, torna-se numa plataforma fechada.

4. Testes e Resultados

Neste capítulo são descritos alguns testes e configurações realizados com os dois sistemas, juntamente com as luzes inteligentes e com os outros dispositivos, para testar as funcionalidades e as capacidades para assim comparar os sistemas.

4.1. Home Assistant

Com este sistema é permitida uma escolha aberta para interagir em qualquer sistema operativo, sendo por isso possível fazer os nossos testes sem limites [44, 45].

Os testes foram aplicados para os seguintes sistemas:

- Máquinas virtuais com o sistema operativo Ubuntu 20.04 LTS para as versões Core e Supervised;
- Home Assistant Operating System no Raspberry Pi 3+;
- Máquina virtual com Windows 10;
- Sistema operativo Windows 10 com WSL.

A possibilidade de aplicar estes testes em todos estes sistemas só foi possível graças à documentação existente para *developers* (*Home Assistat Developer Docs*) [88] e à comunidade que fornece as fontes necessárias para configurar estas três versões do Home Assistant [89].

4.1.1. Home Assistant Core

A versão Core, como foi mencionado na secção 3.2.1, é um programa Python que executa as funcionalidades essenciais da aplicação Home Assistant.

A instalação do Core pode ser feita de duas formas distintas:

- Aplicar *fork* do repositório oficial, como foi mencionado na secção 3.2.8;
- Instalar manualmente no terminal;

Na segunda forma de instalar do sistema, devemos implementar um conjunto de comandos essenciais: atualizar o sistema, instalar as dependências e criar um ambiente virtual do Home Assistant Core, como podemos ver nas Figuras 55, 56, 57 e 58.

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

Figura 55. Atualização do sistema operativo [44]

```
sudo apt-get install -y python3 python3-dev python3-venv python3-pip libffi-dev
libssl-dev libjpeg-dev zlib1g-dev autoconf build-essential libopenjp2-7 libtiff5
libturbojpeg0 tzdata
```

Figura 56. Adicionar dependências [44]

```
sudo -u homeassistant -H -s
cd /srv/homeassistant
python3.9 -m venv .
source bin/activate
```

Figura 57. Criar o ambiente virtual [45]

```
python3 -m pip install wheel
```

```
pip3 install homeassistant
```

```
hass
```

Figura 58. Instalar e executar HA [45]

Refira-se que estes comandos também funcionam nos subsistemas Linux que são criados nos sistemas operativos Windows 10 e Windows 11.

Se for para utilizar o programa no Windows 10, podemos recorrer as duas formas distintas de instalação: instalar manualmente os comandos no terminal de Python ou utilizar um instalador feito pelo utilizador AlexxIt [96];

A primeira opção é simples, contudo, por vezes pode falhar a instalação do ambiente virtual no sistema Windows. Os comandos de Python seguem o mesmo princípio como acontece no Linux.

Na segunda opção, o Home Assistant Windows Portable (ou HassWP) [96], instala alguns programas necessários, incluindo o Python, e aparece logo uma consola para executar as funcionalidades do programa, como vemos na Figura 59.

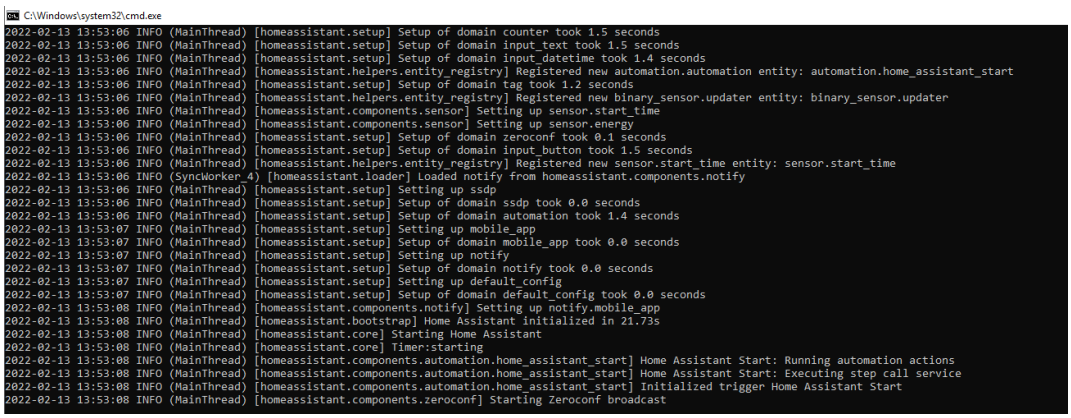

Figura 59. Execução do HAssWP

A vantagem de recorrer a este método é que facilita o uso da aplicação e não é necessário implementar os comandos como acontece no sistema Linux.

O Home Assistant Core tem a capacidade de instalar integrações e verifica a existência de dispositivos que estejam presentes na rede de casa, contudo, não existe a possibilidade de implementar extensões no sistema [97], como foi mencionado na secção 3.2.1.

4.1.2. Home Assistant Supervised

Esta versão tem apenas duas opções de instalação: instalar o sistema Linux numa máquina virtual ou instalar *bare metal,* o sistema operativo Linux, para executar a versão Supervised [89].

Como foi referido anteriormente na secção 3.2.1, esta versão não é recomendada para utilizadores inexperientes. Portanto, deve ser instalada com precaução [89, 90].

A parte essencial em que devemos ter precaução é a virtualização com o Docker. Quando instalamos o programa através do Docker, este executa automaticamente a virtualização do Home Assistant [89]. Ao contrário do Home Asssistant Core, esta versão tem a vantagem de não exigir uma ativação manual.

Para além de adicionar as integrações, temos também a possibilidade de adicionar extensões oficiais do Home Assistant (Duck DNS, Mosquito Broker) e, ainda, podemos adicionar extensões feitas pela comunidade (Node-Red, Pyscript, Visual Studio Code).

Nos ensaios realizados foram usadas as extensões: ESPHome, Visual Studio Code, Samba Share e Terminal SSH.

Após a instalação da extensão básica *"Hello World"*, como foi mencionado na secção 3.2.10.2, surgiram episódios de instabilidade, levando à conclusão de que o sistema não é capaz de instalar extensões ou configurar por não estar "saudável". Este é um dos motivos principais que os utilizadores devem prestar atenção quando interagem com esta versão, pois devem sempre atualizar o sistema operativo.

Também surgiu instabilidade com o Docker devido às falhas de autorização para aceder ao Home Assistant. Para resolver este problema, teve de ser adicionada uma autorização pelo Docker. Caso permanecesse a mesma falha para aceder ao Home Assistant, tinham que ser reinstalados o Home Assistant Supervised e o Docker.

4.1.3. Home Assistant OS

A versão OS é considerada a versão mais estável. Tem uma instalação mais fácil que as outras versões, pois basta instalar o sistema num cartão SD e configurar a rede Ethernet (endereço IP). Depois de serem aplicadas essas configurações, o sistema faz o resto da instalação.

O Home Assistant Operating System tem um sistema operativo Linux mínimo, mencionado na secção 3.2.11, sendo que a única forma para interagir com o sistema, de forma remota, passa pela inserção de extensões ou integrações, nomeadamente através das seguintes extensões: Terminal SSH, Samba Share e Visual Studio Code.

Uma das vantagens principais deste tipo de sistema é que não exige a dependência do utilizador para atualizar o sistema. Como tem a terceira camada do Home Assistant, tem uma melhor gestão para atualizar ou para efetuar backup, no caso da nova versão ficar corrompida.

Com a versão Operating System foram realizados os testes com os dispositivos que foram mencionados na secção 3.2.

4.1.4. Experiências com os dispositivos

Como o Home Assistant é *open-source*, notamos que tem grande suporte por parte da comunidade e dos fabricantes e torna-se simples implementar qualquer dispositivo possível no sistema.

Para começar, foram instaladas as extensões para interagir com o Home Assistant, como foi mencionado no capítulo anterior. Também foram instaladas as integrações e extensões para configurar os dispositivos, nomeadamente a extensão ESP Home e a integração Philips Hue.

Para implementar o sistema ESP Home existe um conjunto de instruções para configurar o dispositivo ESP8266 num ficheiro YAML, de forma a apresentar os sensores de humidade e de temperatura, como podemos ver na Figura 60. Após as configurações iniciais, tem de ser compilado um ficheiro binário para executar no Home Assistant.

O resultado final desta extensão expõe a informação dos sensores de temperatura e de humidade e atualiza os dados de cinco em cinco minutos, como podemos ver na Figura 61.

```
esphome:
  name: living_room_nodemcu_temphum_sensor
  platform: ESP8266
  board: nodemcuv2

wifi:
  ssid: "    "
  password: "    "

  # Enable fallback hotspot (captive portal) in case wifi connection fails
  ap:
    ssid: "Nodemcu1 Fallback Hotspot"
    password: "mypass12"

captive_portal:

# Enable logging
logger:

# Enable Home Assistant API
api:
  password: "mypass"

ota:
  password: "mypass"

sensor:
  - platform: dht
    pin: D6
    temperature:
      name: "Living Room Temperature"
    humidity:
      name: "Living Room Humidity"
    update_interval: 30s
    model: AM2302
```

Figura 60. Configuração dos sensores no ESP

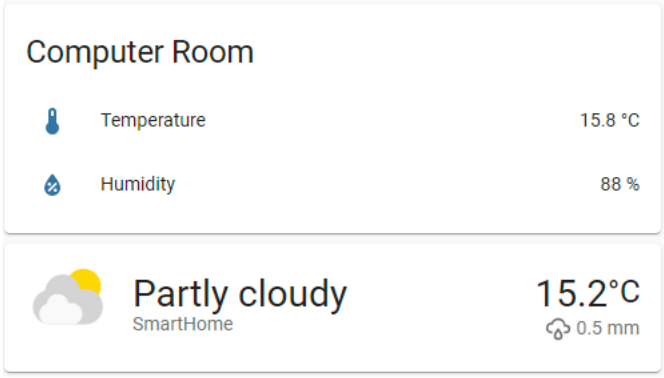

Figura 61. ESP configurado

Para os dispositivos do fabricante Philips, foi necessário instalar a aplicação Philips Hue, disponível para Android e iOS, de maneira a adicionar os dispositivos inteligentes juntamente com a Philips Hue Bridge. Depois da instalação dos dispositivos, passamos a instalar a integração da Philips Hue.

Primeiro procuramos os endereços IP da Philips Hue Bridge (Figura 62) e depois sincronizamos todos os dispositivos que estejam ligados nessa ponte.

Figura 62. Deteção do Hue Bridge

Figura 63. Configuração do Philips Hue

Após a configuração no sistema Home Assistant (Figura 63), foi possível testar e experimentar a integração da Philips Hue, sem voltar à aplicação móvel Philips Hue.

A integração da Philips Hue simplifica a interação com as duas luzes, ativando e desativando as luzes ou alterando a intensidade e a temperatura das luzes, como podemos ver nas Figuras 64 e 65.

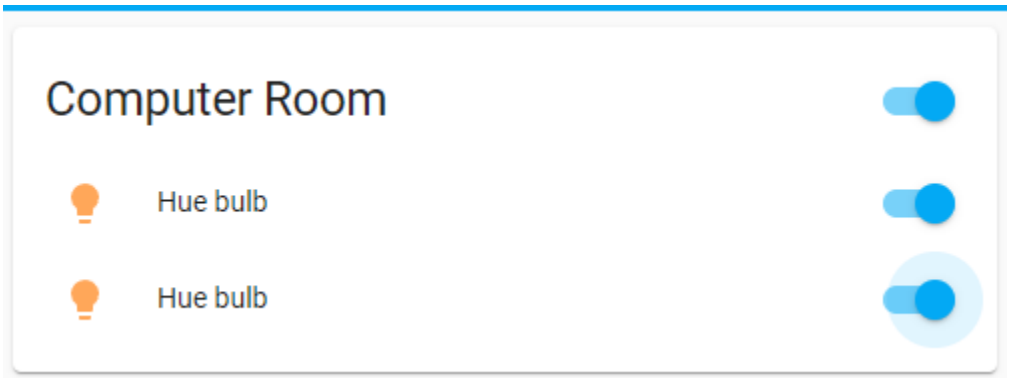

Figura 64. Philips Hue no Painel do HA

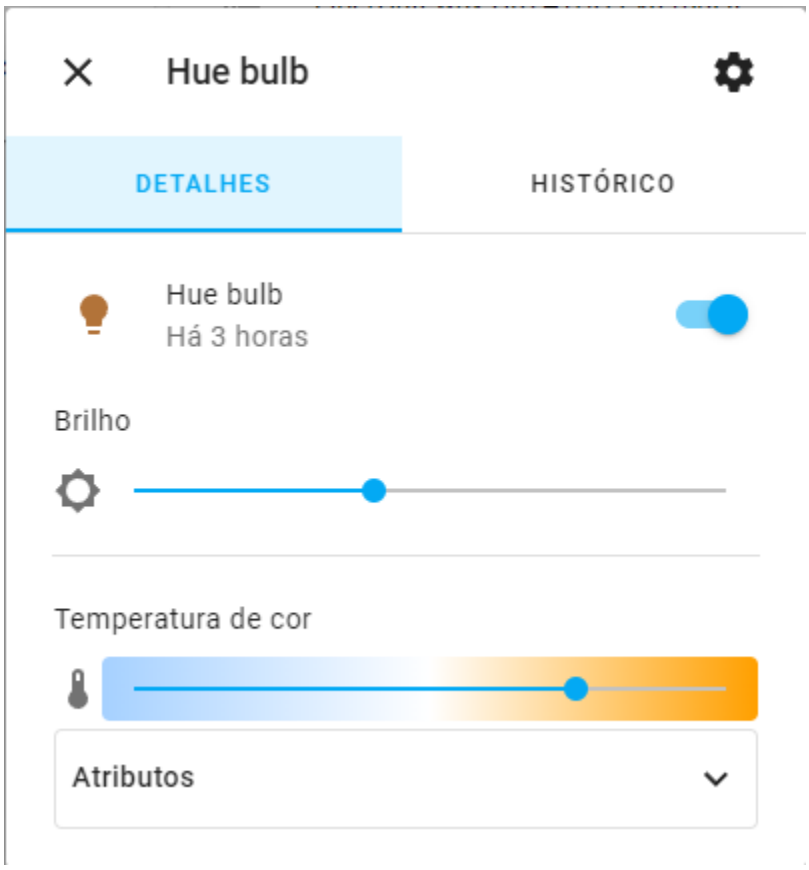

Figura 65. Configuração da lâmpada

Opcionalmente, pode-se criar um conjunto de cenários para automatizar as luzes, como por exemplo manipular ou ativar as luzes.

Primeiro temos que configurar o ficheiro para configurar um conjunto de luzes, como podemos ver na Figura 66. Neste caso escolhemos as entidades das duas luzes, descritas no ficheiro o 'ligths.yaml'.

```
lights.yaml
1  - platform: group
2    name: Lamps
3    entities:
4      - light.hue_bulb
5      - light.hue_bulb_2
```

Figura 66. Configuração das luzes

Se o utilizador tiver mais do que 2 luzes Philips, ou de outro fabricante compatível, pode especificar as luzes que deseja ativar automaticamente através do cenário.

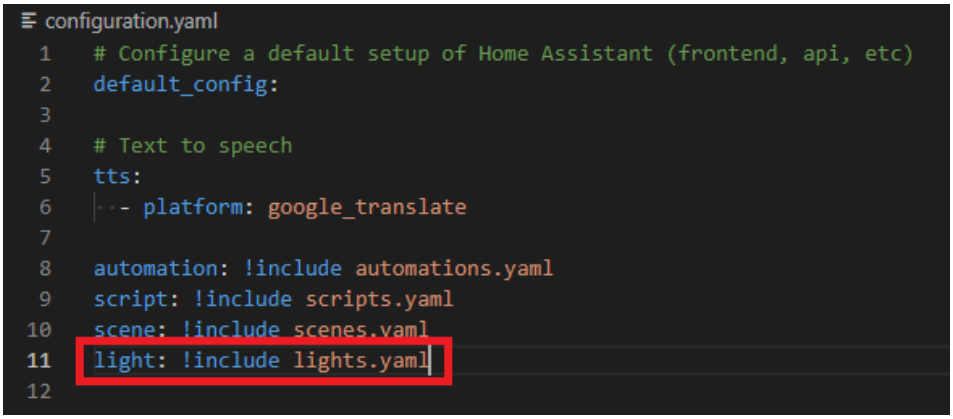

Figura 67. Adição das luzes

De seguida, deve ser invocado o ficheiro das luzes no ficheiro da configuração do Home Assistant, como podemos ver na Figura 67. Para confirmar se está a funcionar, voltamos ao *dashboard* para verificar no sistema se a *syntax* está correta no ficheiro da configuração. Após a confirmação, reiniciamos o sistema. Depois de ser reiniciado, o sistema apresenta, no *dashboard,* as luzes que foram configuradas, como indica na Figura 68.

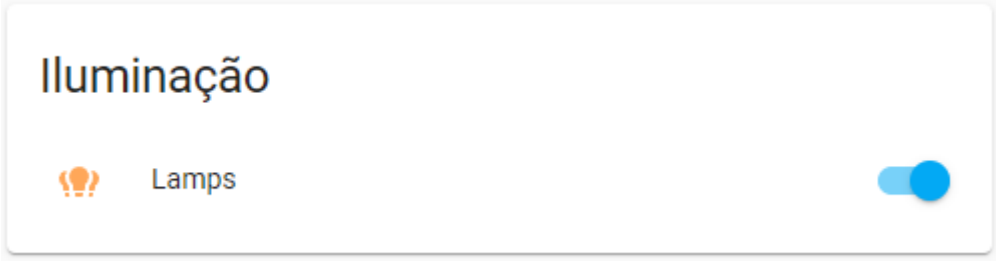

Figura 68. Luzes configuradas

Para criar dois cenários, um para minimizar a intensidade das luzes e outro para intensificar as luzes, devemos agrupar as mesmas luzes configuradas. Posteriormente, estas luzes vão para dentro da configuração do cenário. Depois de agrupar esse conjunto de luzes, manipula-se a intensidade e o brilho das luzes com dois cenários '*Night Scene*' e '*Day Scene*', como podemos ver nas Figuras 69 e 70.

Com este resultado, um utilizador com alguma instrução básica consegue criar um conjunto de cenários na *dashborad* e, efetivamente, obtém os cenários que deseja.

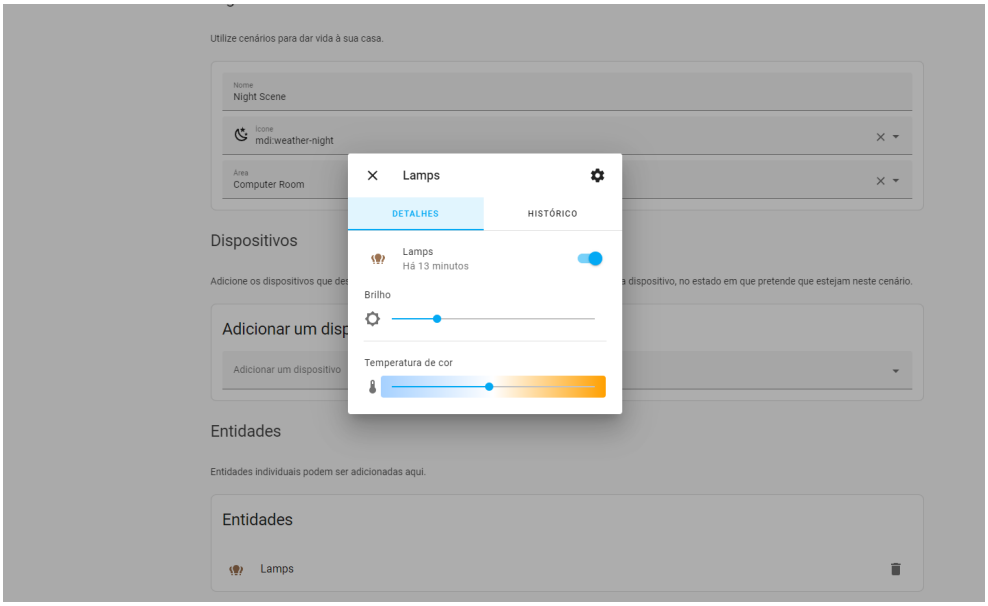

Figura 69. Configuração de um cenário para noite

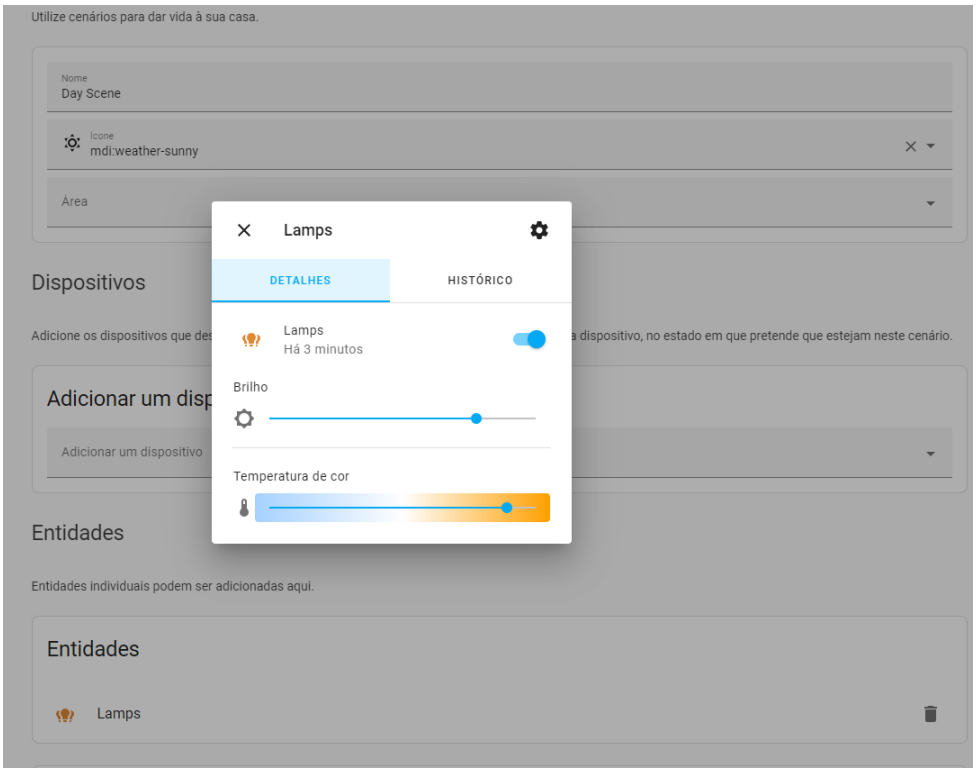

Figura 70. Configuração para o cenário de dia

De seguida procedeu-se à instalação da integração ZHA (*Zigbee Home Automation*) para adicionar as luzes inteligentes, mas não houve sucesso no emparelhamento aos mesmos, devido a limitação na implementação de um novo dispositivo numa rede Zigbee. Esta falha veio chamar a atenção para o facto de ser necessário ter a certeza que os dispositivos têm compatibilidade com as integrações do Home Assistant.

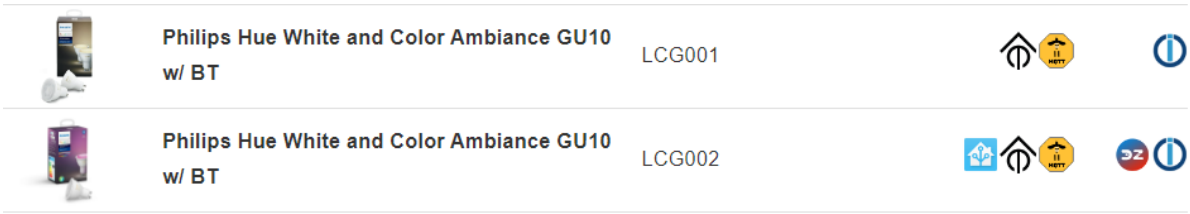

Figura 71. Comparação entre as duas GU10 [98]

Observando a Figura 71 e comparando as duas luzes GU10, verifica-se que o segundo conjunto de luzes apresenta a possibilidade de emparelhar a luz com a integração ZHA, enquanto, o primeiro conjunto de luzes que foram testados, não têm compatibilidade com essa integração. A única opção que tivemos foi explorar com a extensão externa Zigbee2Mqtt.

Na última parte dos testes, para a implementação do MQTT no Home Assistant, foi necessário instalar duas extensões: o Mosquitto Broker e o Zigbee2Mqtt. Ao instalar o Zigbee2Mqtt tivemos que configurar o servidor, sendo que foi necessário adicionar as seguintes partes: o endereço do servidor, credenciais (nome do utilizador e a password) e a porta do *hardware* para a *pendrive* Zigbee.

```
data_path: /config/zigbee2mqtt
socat:
  enabled: false
  master: pty,raw,echo=0,link=/tmp/ttyZ2M,mode=777
  slave: tcp-listen:8485,keepalive,nodelay,reuseaddr,keepidle=1,keepintvl=1,keepcnt=5
  options: '-d -d'
  log: false
mqtt:
  server: mqtt://core-mosquitto
serial:
  port: /dev/ttyUSB0
zigbee_herdsman_debug: false
zigbee_shepherd_devices: false
```

Figura 72. Configuração do MQTT

Na configuração, como vemos na Figura 72, adicionamos ao servidor o 'core-mosquitto', que refere a extensão Mosquitto Broker e mencionamos a porta do *hardware* '/dev/ttyUSB0'. Isto refere-se ao SONOFF que foi implementado no Raspberry Pi. Se fosse para utilizar o Zigbee CC2531 teria que ser mencionada outra porta.

Ao observar a documentação do Zigbee2Mqtt [99], verificou-se que os dois Zigbee que foram utilizados têm compatibilidades diferentes. Comparando esses dois dispositivos, o SONOFF acaba por ser mais recomendável, pois tem uma *firmware* mais recente e tem um grande alcance às redes, enquanto que o Texas CC2531 acaba por não ser recomendável, pois, apesar de ter um suporte para muitos dispositivos, tem um chip desatualizado.

Depois de abrir o servidor, têm de ser configuradas as credencias do utilizador local e a password para ter acesso às informações do MQTT. Para começar a procurar os dispositivos clica-se no botão 'Permit join (All)'.

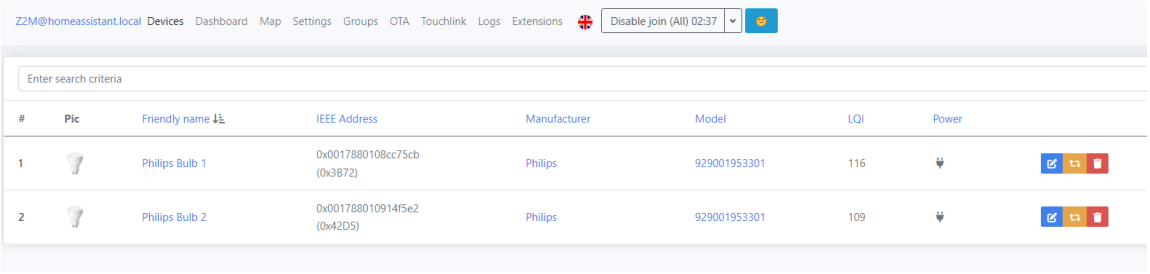

Figura 73. Deteção das luzes

No servidor do Zigbee2Mqtt adicionamos, facilmente, os dispositivos inteligentes na rede MQTT, como vemos na Figura 73. Podemos também configurar o nome dos dispositivos e ajustamos a configuração das luzes, como conseguíamos fazer na integração Philips Hue.

Como nota final, refira-se que o sistema do Home Assistant representa uma abertura para utilizar imensas integrações e extensões que são desenvolvidas pela comunidade. No entanto, o utilizador tem que ter atenção às compatibilidades dos dispositivos com os protocolos e, também, com as integrações do sistema.

4.2. Hubitat

Para iniciarmos as experiências com o Hubitat, não foi precisa nenhuma instalação ou configuração do sistema. Só exigiu a criação de uma conta local para interagir com o sistema.

A instalação das aplicações ou dos dispositivos, não apresentam complexidade mas não existe grande variedade de escolha para configurar no sistema.

4.2.1. Aplicação Hue Bridge Integration

Primeiro, foi feita a instalação da aplicação Hue Bridge Integration, como apresentado na documentação do Hubitat [92]. Ao adicionar a integração, é realizada a procura dos dispositivos Hue Bridge, que fornecem a informação do Bridge que foi encontrado na rede, como representamos na Figura 74.

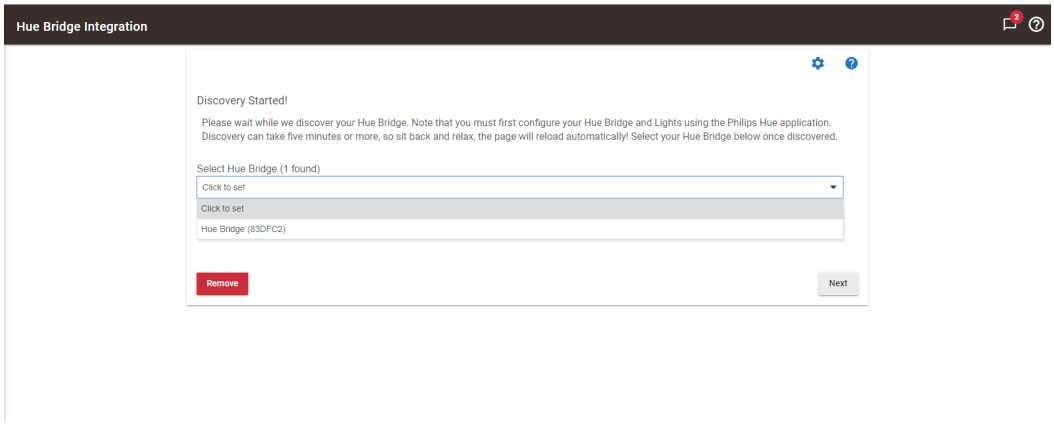

Figura 74. Procura do Hue Bridge

Após a escolha do dispositivo Hue Bridge, é nos dada a informação para esperar e atualizar as informações do aparelho. Contudo, verificou-se que foi necessário realizar um *refresh* à página do Hubitat para passar para o próximo passo da integração Hue Bridge.

Bastou adicionar um *check mark* às luzes que foram encontradas, sendo que, opcionalmente, existem outras funcionalidades que podemos configurar, como por exemplo encontrar grupos, conforme podemos ver na Figurar 75. Se existirem outros Hue Bridge na rede da casa será necessário repetir estes passos.

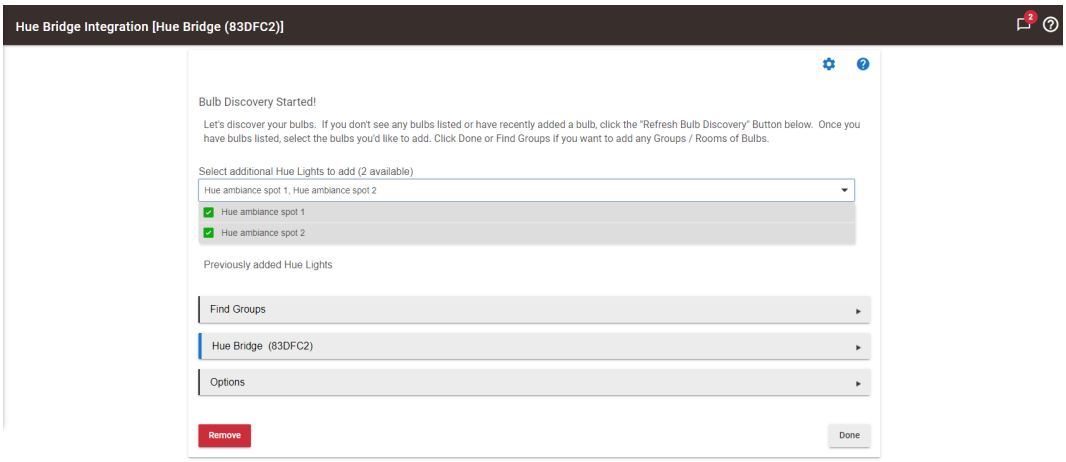

Figura 75. Encontrando as luzes Hue

4.2.2. Configuração dos dispositivos

Na configuração dos dispositivos existentes no Hubitat, cada dispositivo é configurado numa página com os inputs incluídos para configurar esse dispositivo. Esta configuração dos dispositivos revelou-se o primeiro problema do Hubitat, pois não é fornecida uma informação explícita ou *user-friendly* para o utilizador. Por exemplo, na configuração das luzes utilizadas,

é possível configurar a sua cor entre o frio e o quente, e esta integração não tem informação visual para determinar a intensidade da luz e a temperatura da mesma.

Para um utilizador normal, que pode não ter conhecimento quando insere o valor da temperatura da luz, os valores que são apresentados com a medida kelvin, e não existe *feedback* para confirmar se foram efetuadas as mudanças, apenas possuímos a confirmação de mudança por parte do Hubitat, se estivermos nas proximidades dos dispositivos.

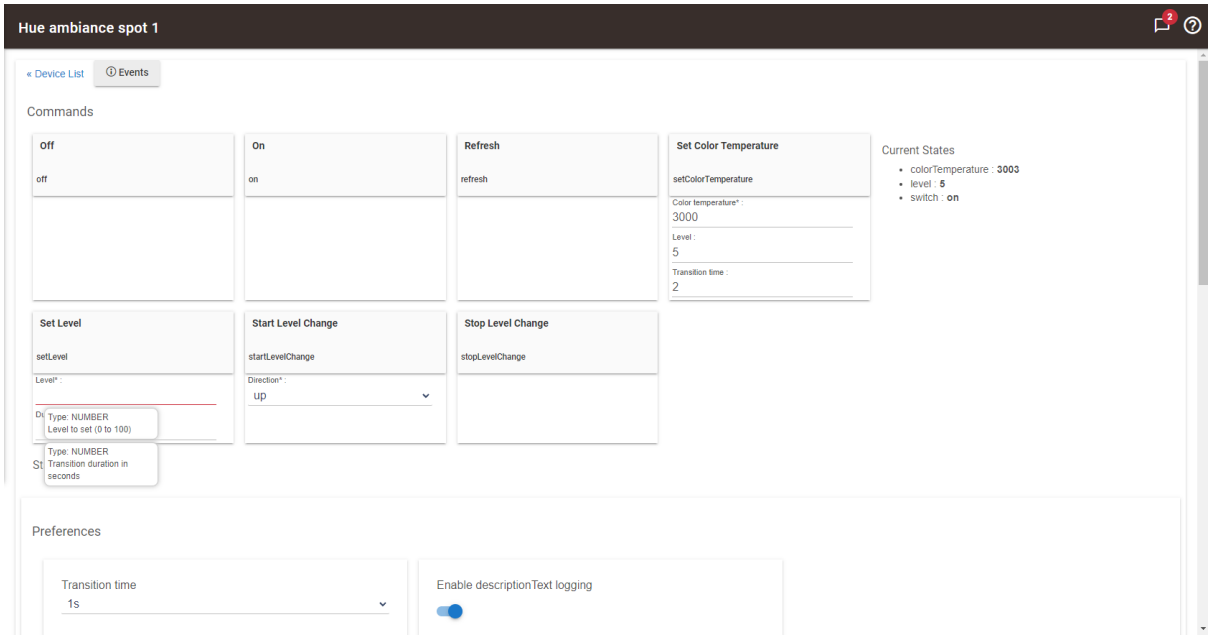

Figura 76. Configuração da luz

Em resumo pode-se referir que o *front-end* destas páginas não está bem configurado, pois quando interagimos com os inputs, não é devolvido o *feedback,* na página.

4.2.3. Configuração para uma regra

Existem dois tipos de automações: a *Basic Rule* e a *Rule Machine*. Foi primeiro usado o modo *Basic Rules* para configurar um cenário básico. Este não tem grande complexidade para automatizar os dispositivos que são selecionados. Caso o utilizador precise de instruções de como configurar uma regra básica, existe um vídeo explicativo de como se deve configurar os dispositivos.

Dentro da regra básica, existem as seguintes opções para configurar: interagir com um *switch*, o tempo do dia e o modo do sistema. Neste caso, como vemos na Figura 77, adicionamos a opção do tempo, do dia e às dezasseis horas e vinte e cinco minutos, as luzes são acionadas e a temperatura e o nível das lâmpadas são configurados.

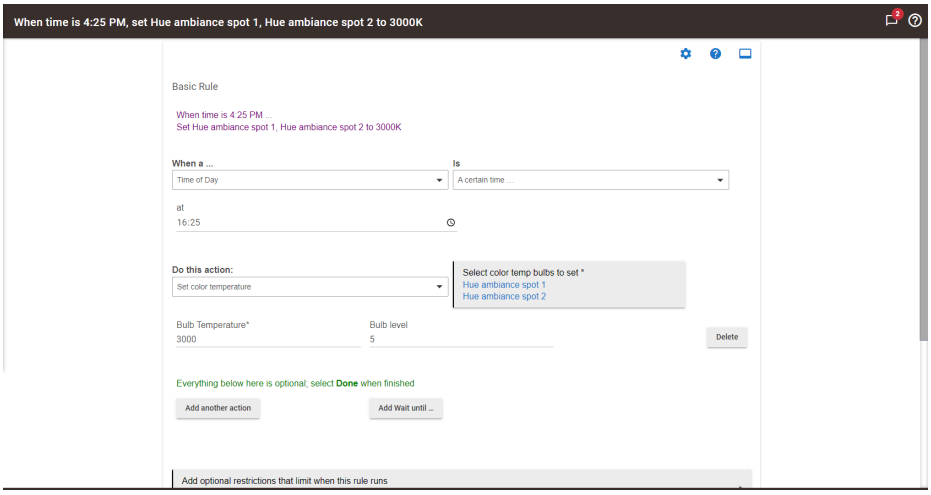

Figura 77. Regra básica das luzes

A configuração desta automatização tem uma grande vantagem, pois se for um utilizador que não tenha experiência em programação, não precisa de configurar uma linha de código para criar este tipo de automações.

Caso o utilizador desejar usar o modo de automação *Rule Machine*, tem de implementar uma lógica para recorrer a essas regras, como podemos observar na Figura 78, em que é configurada uma condição para ativar as luzes depois do por do sol.

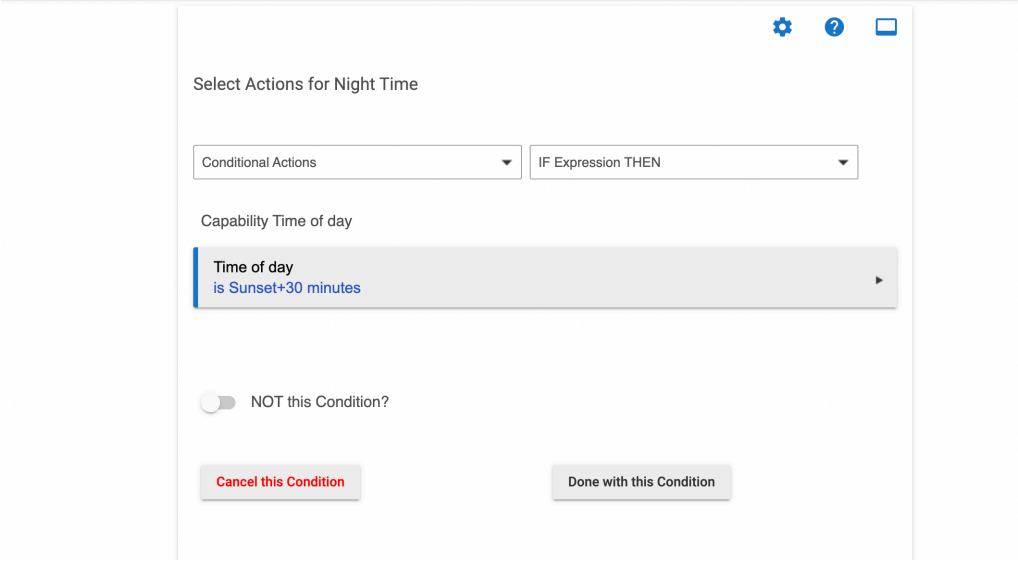

Figura 78. Regra condicional com Rule Machine

Se o utilizador pretender aprofundar mais este tipo de regras tem de programar com a linguagem de programação Grovy e adicionar os dispositivos que deseja configurar. Neste caso, usando os dispositivos Philips Hue teria que implementar um conjunto de passos para configurar os drivers dos dispositivos para interagir com o *Rule Machine*.

4.2.4. Adicionar os dispositivos manualmente

A outra opção para adicionar os dispositivos é selecionar a opção *default* do Hubitat, como foi mencionado na secção 3.3.2, com os protocolos Zigbee e Z-Wave pré-implementados. Como referido na documentação do Hubitat, basta clicar no protocolo que desejamos e adicionar manualmente.

Após a escolha do protocolo Zigbee, é ativado o modo de descoberta dos dispositivos e encontram-se os mesmos, facilmente, em alguns segundos. No registo dos *logs* do Hubitat observa-se o tempo durante o qual decorreu a descoberta dos aparelhos, como podemos ver na Figura 79.

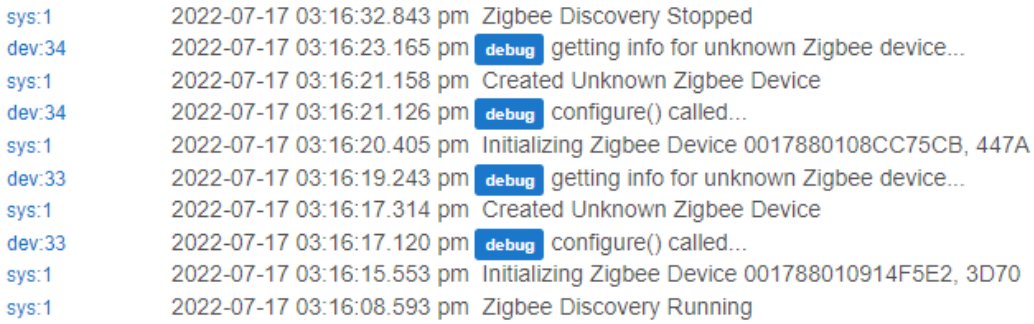

Figura 79. Registo durante a descoberta dos dispositivos

Como nota final refira-se que com este sistema a principal limitação encontra-se na configuração dos dispositivos, que não tem informação para configurar as luzes nem tem um bom feedback ao interagir com a aplicação. Ao contrário da integração do Hue Bridge, a configuração manual torna-se mais árdua para o utilizador.

4.3. Comparação final entre Home Assistant e Hubitat

Para finalizar a comparação entre os dois sistemas: Home Assistant e Hubitat, refira-se resumidamente que o Hubitat centra no mercado, enquanto, o Home Assistant centra-se nos *geeks* ou entusiastas [100].

Nestas comparações finais avaliamos e analisamos as comparações que foram feitas pelo autor [100] nos seguintes temas:

- Vantagens e desvantagens;
- Automações;
- Protocolo Zigbee;
- Escolha do sistema.

4.3.1. Vantagens e desvantagens

Cada sistema tem as suas capacidades e as suas características únicas, ideais para qualquer utilizador. Para motivar o interesse e a escolha entre os sistemas Home Assistant e Hubitat, foram exploradas as vantagens e as desvantagens destes sistemas.

Começando pelo sistema Hubitat, este tem uma grande comunidade de *developers*, mas desconhecemos se tantos como a comunidade do Home Assistant, e suporta assistentes virtuais. Contudo, o sistema Hubitat é uma plataforma fechada, não tem uma boa interface, nem fornece informação explicita ou feedback para os utilizadores e tem uma documentação é limitada.

No caso do sistema Home Assistant, como foi mencionado na secção 3.2, este tem as vantagens de ser um sistema *open source*, tem uma boa interface e tem uma boa documentação para qualquer *developer*.

Em termos de desvantagens: o sistema não tem uma interface *user-friendly* e exige alguma prática. Além disso, as integrações e as extensões têm instalações complexas e instalar os assistentes virtuais implica uma subscrição mensal ou tem de ser programado.

4.3.2. Automação

Na parte da automação existem algumas diferenças entre estes dois sistemas, quando o utilizador cria uma automação entre estes dois sistemas.

Com o Hubitat torna-se mais fácil de implementar as aplicações e as automações do que com o sistema Home Assistant.

Assistentes de voz como a Alexa e o Google Assistant podem ser implementados no Hubitat, contudo, os assistentes de voz não têm funcionalidade remota. A solução seria implementar uma VPN e adicionar alguma programação.

Para os utilizadores que estejam a mexer pela primeira vez, julgamos que estes conseguem criar automações básicas facilmente com o modo *Basic Rules*, mas se for para aprofundar a automação terão de utilizar o modo *Rule Machine*.

O sistema Home Assistant tem uma boa capacidade de gestão e tem maior compatibilidade de dispositivos, quando comparando com o Hubitat.

Apesar de ter maior compatibilidade com os produtos inteligentes, o sistema Home Assistant exige alguma prática para configurar e programar as automações.

Para a utilização de assistentes de voz existem duas opções: fazer uma subscrição mensal ou programar os assistentes de voz que o utilizador deseja instalar. Se for para trabalhar

remotamente exige algum desenvolvimento interno, especialmente para integração dos protocolos Zigbee e Z-Wave.

4.3.3. Protocolo Zigbee

Durante as experiências, houve a possibilidade de comparar o protocolo Zigbee nos sistemas Home Assistant e Hubitat.

No sistema Home Assistant, o utilizador tem a possibilidade de configurar qualquer integração ou extensão para utilizar o protocolo Zigbee. Contudo, o utilizador tem que planear e pesquisar se os dispositivos são compatíveis com a integração ou extensão que pretende instalar no Home Assistant, pois nem sempre é possível instalar qualquer dispositivo nas integrações e nas extensões.

Para o sistema Hubitat, não há uma grande variedade de escolha para instalar integrações para utilizar o protocolo Zigbee. No entanto, as que estão disponíveis são mais fácies, como aconteceu na instalação da integração do Philips Hue Brigde. Estas integrações têm a vantagem de fornecer melhor informação do que a instalação manual do Hubitat, pois a interface não tem informação suficiente.

4.3.4. Limitações

Antes de testar os dois sistemas, os dispositivos inteligentes da Philips têm de, em primeiro lugar, ser emparelhados com a aplicação móvel da Philips Hue, que estão disponíveis nos sistemas Android e iOS. No entanto, verificou-se que falhou o emparelhamento dos dispositivos na aplicação móvel para ambos os sistemas móveis e estes aparelhos tiveram de ser emparelhados manualmente.

Exploraram-se, também, os protocolos de comunicação, o Zigbee e o MQTT, que foram testados com os dispositivos inteligentes do fabricante Philips. Durante a fase dos testes, estes protocolos apresentaram limitações. Quando foram emparelhados os dispositivos com as integrações do Home Assistant e do Hubitat, descobriu-se que nem todos os dispositivos são compatíveis nas integrações dos sistemas.

4.3.5. A escolha do sistema

A escolha entre os dois sistemas Home Assistant e Hubitat variam devido às suas capacidades e às necessidades do utilizador. O autor da referência [100], aponta a sua

preferência ao Hubitat, por ter menor complexidade na configuração dos dispositivos que o sistema Home Assistant.

Nesta escolha final entre os sistemas Home Assistant e o Hubitat, fornecemos outro ponto de vista entre estes dois sistemas.

O sistema Home Assistant, representa a maturidade e o desenvolvimento contínuo de um sistema para casas inteligentes. Julgamos por isso, que tornar-se-ia na melhor escolha para qualquer utilizador, se não tivesse a complexidade de configurar e de programar os dispositivos.

Caso o utilizador não deseje muita complexidade ou programação, o Hubitat é um sistema ideal. A parte da programação torna-se opcional nalguns pontos de desenvolvimento no sistema do Hubitat e exige alguma prática. O utilizador familiariza-se com essa programação se mexeu anteriormente no sistema SmartThings. No entanto, o sistema Hubitat não devolve bom *feedback* e não apresenta informação explicita aos utilizadores. O sistema Hubitat continua ainda em crescimento e promete o aumento de compatibilidade e a facilidade de desenvolvimento dos dispositivos no mercado das casas inteligentes.

5. Conclusão

Esta tese explora, resumidamente, os conceitos da Internet das Coisas e das casas inteligentes. Ao integrar-se o conceito das casas inteligentes na Internet das Coisas fez com que se evoluísse o desenvolvimento deste conceito. Nota-se que o conceito das casas inteligentes se adapta, cada vez mais, a qualquer casa. Atualmente, não exige o desenvolvimento de um *hardware* dispendioso e que se torna imóvel, como acontecia há quarenta anos. A instalação tornou-se mais fácil e podemos utilizar placas únicas (por exemplo o Raspberry Pi) ou reutilizar *hardware* antigo para a rede de uma casa inteligente.

O problema principal para os desenvolvedores é criar um sistema operativo especializado para múltiplos dispositivos inteligentes de diversos fabricantes. Os dois sistemas estudados nesta tese tentam resolver o problema principal de um sistema operativo para casas inteligentes.

Este trabalho consistiu, principalmente, na comparação entre os sistemas Home Assistant e Hubitat, analisando as documentações e as funcionalidades dos mesmos. Ambos têm caraterísticas diferentes que motivam os utilizadores como: a simplicidade, a interface do utilizador, a automatização e a compatibilidade com diferentes dispositivos.

Após a realização dos testes, apresentamos alguns pontos de reflexão caso o utilizador deseje escolher, entre um destes dois sistemas. O sistema Hubitat apresenta alguns pontos que demostram a simplicidade de configurar e de automatizar os dispositivos inteligentes. Por outro lado, o sistema Home Assistant aprofunda a compatibilidade dos dispositivos de muitos fabricantes e o desenvolvimento das integrações e de extensões graças à comunidade do Home Assistant.

5.1. Limitações deste trabalho

Deve ser mencionado, que na fase de pesquisa foram sentidas dificuldades em encontrar documentos ou exemplos de estudos de comparação entre os sistemas do Home Assistant e do Hubitat.

Nesse aspeto o sistema Home Assistant, é um pouco mais mencionado, nalguns documentos académicos, em que os investigadores o utilizaram para estudar os protocolos (por exemplo: MQTT) ou comparar com o sistema OpenHab.

No caso do sistema Hubitat, por ser um *software closed source,* existem poucos exemplos de documentação para estudar a funcionalidade do sistema. Existem alguns *sites* que comparam estes dois sistemas, feitos por entusiastas de casas inteligentes, mas a documentação do sistema não aprofunda o conteúdo do mesmo.

5.2. Trabalho futuro

Neste estudo, nota-se que existe a possibilidade de desenvolver projetos para aprofundar os dois sistemas: Home Assistant e Hubitat. Possivelmente, nos próximos projetos poderão ser desenvolvidas integrações ou extensões customizadas.

Ambos os sistemas têm pontos interessantes, tais como o desenvolvimento de um sistema operativo e o alcance do objetivo de confortar o utilizador. O Home Assistant aproxima os entusiastas de casas inteligentes e o Hubitat aproxima os utilizadores que desejam a simplicidade.

Estes e outros sistemas estão-se a tornar, cada vez mais, *user-friendly*, especialmente nas automações e na possibilidade de operarem com dispositivos (e protocolos) diferentes. Possivelmente, os próximos sistemas para casas inteligentes estarão aptos para qualquer utilizador, sem restringir o conhecimento tecnológico. Por outras palavras, cada vez mais teremos sistemas à base de *low-code* e *no-code*.

Referências

[1] *What is IoT (Internet of Things) and How Does it Work?* (n.d.). Retrieved April 8, 2021, from https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

[2] *The History of the Internet of Things*. (n.d.). Retrieved November 29, 2020, from https://perenio.com/blog/the-history-of-the-internet-of-things

[3] *A Brief History of the Internet of Things - DATAVERSITY*. (n.d.). Retrieved November 29, 2020, from *https://www.dataversity.net/brief-history-internet-things/*

[4] *The little-known story of the first IoT device - Industrious*. (n.d.). Retrieved October 24, 2021, from https://www.ibm.com/blogs/industries/little-known-story-first-iot-device/

[5] Kevin Asthon. (2010). That ' Internet of Things ' Thing. *RFID Journal*, 4986. http://www.rfidjournal.com/article/print/4986

[6] Jiang, L., Liu, D. Y., & Yang, B. (2004). Smart home research. *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, *2*(August), 659–663. https://doi.org/10.1109/icmlc.2004.1382266

[7] Zouai, M., Kazar, O., Haba, B., & Saouli, H. (2017). Smart house simulation based multi-agent system and internet of things. *Proceedings of the 2017 International Conference on Mathematics and Information Technology, ICMIT 2017*, *2018-Janua*, 201–203. https://doi.org/10.1109/MATHIT.2017.8259717

[8] *Complete Smart Home App Development Guide for Entreprenuers 2020*. (n.d.). Retrieved October 29, 2020, from https://www.unifiedinfotech.net/blog/smart-home-app-development-guide/

[9] *The Evolution of Smart Home Technology*. (n.d.). Retrieved November 29, 2020, from https://blog.bccresearch.com/the-evolution-of-smart-home-technology

[10] Lytvyn, V., Vysotska, V., Mykhailyshyn, V., Peleshchak, I., Peleshchak, R., & Kohut, I. (2019). Intelligent system of a smart house. *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019 - Proceedings*, 282–287. https://doi.org/10.1109/AIACT.2019.8847748

[11] Tulenkov, A., Parkhomenko, A., Sokolyanskii, A., Stepanenko, A., & Zalyubovskiy, Y. (2018). The features of wireless technologies application for smart house systems. *Proceedings of the 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems, IDAACS-SWS 2018*, 1–5. https://doi.org/10.1109/IDAACS-SWS.2018.8525842

[12] *10 Smart Home App That'll Make Your Life Easier In 2020*. (n.d.). Retrieved November 3, 2020, from https://www.unifiedinfotech.net/blog/top-smart-home-apps/

[13] *Hubitat Elevation™ | Local, Reliable, Fast and Private Home Automation – HUBITAT*. (n.d.). Retrieved January 17, 2021, from https://hubitat.com/

[14] Helal, S., & Chen, C. (2009). *The Gator Tech Smart House*. 1. https://doi.org/10.1145/1592700.1592715

[15] Sanchez, V. G., Pfeiffer, C. F., & Skeie, N. O. (2017). A review of smart house analysis methods for assisting older people living alone. *Journal of Sensor and Actuator Networks*, *6*(3), 1–38. https://doi.org/10.3390/jsan6030011

[16] *TRON Intelligent House*. (n.d.). Retrieved January 28, 2021, from http://tronweb.super-nova.co.jp/tronintlhouse.html

[17] *House of Controversy : Technology: A new computer standard automates the TRON home with 1,000 microprocessors. Some American firms, however, are leery of Japanese intentions. - Los Angeles Times*. (n.d.). Retrieved January 28, 2021, from https://www.latimes.com/archives/la-xpm-1990-07-02-fi-598-story.html

[18] *What exactly is the Internet of Things (IoT)? | Grinn Global*. (n.d.). Retrieved August 3, 2021, from https://grinn-global.com/news-blog/what-exactly-is-the-internet-of-things-iot/

[19] *Kevin Ashton Invents the Term "The Internet of Things" : History of Information*. (n.d.). Retrieved April 8, 2021, from https://www.historyofinformation.com/detail.php?id=3411

[20] *The History of Smart Homes*. (n.d.). Retrieved August 9, 2021, from https://www.afcdud.com/fr/smart-city/422-how-the-history-of-smart-homes.html

[21] Bradbury, R. (1950). There Will Come Soft Rains. *The Martian Chronicles*, 206–211.

[22] *Smart Home Technology in 1966 - Smart Homes & Smart Offices*. (n.d.). Retrieved August 4, 2021, from https://smartofficesandsmarthomes.com/smart-home-technology-1966/

[23] *Bluetooth for IoT*. (n.d.). Retrieved March 31, 2021, from https://www.aeris.com/news/post/bluetooth-for-iot/

[24] *Introduction to IoT | What is Bluetooth*. (n.d.). Retrieved August 9, 2021, from https://www.leverege.com/iot-ebook/iot-bluetooth

[25] *What Is Wi-Fi? - Definition and Types - Cisco*. (n.d.). Retrieved August 13, 2021, from https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html

[26] *Introduction to IoT | What is WiFi*. (n.d.). Retrieved August 10, 2021, from https://www.leverege.com/iot-ebook/iot-wifi

[27] *The Role of WiFi in IoT*. (n.d.). Retrieved August 10, 2021, from https://www.iotforall.com/wifi-role-iot

[28] *What Is Zigbee Wireless Mesh Networking? | Digi International*. (n.d.). Retrieved April 2, 2021, from https://www.digi.com/solutions/by-technology/zigbee-wireless-standard

[29] *Zigbee 3.0 – Use the Zigbee protocol for IoT solutions*. (n.d.). Retrieved August 18, 2021, from https://www.develcoproducts.com/gateways/wireless-protocols/zigbee/zigbee-30/

[30] *Zigbee Alliance changes name in Matter IoT launch*. (n.d.). Retrieved August 21, 2021, from https://www.eenewseurope.com/news/zigbee-alliance-matter-iot-launch

[31] *Z-Wave Explained: What It Is and How It Works - DIY Smart Home Solutions*. (n.d.). Retrieved April 4, 2021, from https://www.diysmarthomesolutions.com/z-wave-wireless-networking/

[32] *Better and safer smart homes are built on Z-Wave - Z-Wave*. (n.d.). Retrieved August 21, 2021, from https://www.z-wave.com/

[33] *Z-Wave Global Regions - Silicon Labs*. (n.d.). Retrieved August 21, 2021, from https://www.silabs.com/wireless/z-wave/global-regions

[34] *Z Wave Vs ZigBee: Which Is Better For Your Smart Home?* (n.d.). Retrieved August 21, 2021, from https://thesmartcave.com/z-wave-vs-zigbee-home-automation/

[35] A. Sikora, P. Villalonga, e K. Landwehr, «Extensions to Wireless M-Bus Protocol for Smart Metering and Smart Grid Application», em Proceedings of the International Conference on Advances in Computing, Communications and Informatics, New York, NY, USA, 2012, pp. 399–404

[36] *5 Best Smart Home Hubs: SmartThings vs Home Assistant vs HomeKit + More*. (n.d.). Retrieved January 2, 2022, from https://smarthomesolver.com/reviews/5-best-smart-home-hubs/

[37] *Local control vs cloud control - What fits your smart home better? - Yeti Blog*. (n.d.). Retrieved January 9, 2022, from https://getyeti.webflow.io/posts/local-control-vs-cloud-control-what-fits-your-smart-home-better

[38] *Can Your Smart Home Live Without the Cloud?* (n.d.). Retrieved January 15, 2022, from https://homealarmreport.com/smart-home/smart-devices-work-locally/

[39] *What is a Home Automation System? - Definition from Techopedia*. (n.d.). Retrieved April 8, 2021, from https://www.techopedia.com/definition/29999/home-automation-system

[40] *What Is Home Automation and How Does it Work? | Xfinity*. (n.d.). Retrieved August 22, 2021, from https://www.xfinity.com/hub/smart-home/home-automation

[41] *What Is Home Automation and How Does It Work?* (n.d.). Retrieved April 11, 2021, from https://www.security.org/home-automation/

[42] Ringel, M., Laidi, R., & Djenouri, D. (2019). Multiple benefits through smart home energy management solutions—a simulation-based case study of a single-family-house in Algeria and Germany. *Energies*, *12*(8), 1–21. https://doi.org/10.3390/en12081537

[43] *Home Assistant*. (n.d.). Retrieved August 29, 2021, from https://www.home-assistant.io/

[44] *Home Assistant for newcomers: What it is, what is hassio, hassos, hassbian, 101 and cookies - Home Assistant Community*. (n.d.). Retrieved August 29, 2021, from https://community.home-assistant.io/t/homeassistant-for-newcomers-what-it-is-what-is-hassio-hassos-hassbian-101-and-cookies/123004

[45] *Installation - Home Assistant*. (n.d.). Retrieved August 29, 2021, from https://www.home-assistant.io/installation/

[46] *openHAB*. (n.d.). Retrieved August 29, 2021, from https://www.openhab.org/

[47] Saxena, S., Jain, S., Arora, D., & Sharma, P. (2019). Implications of MQTT connectivity protocol for iot based device automation using home assistant and OpenHAB. *Proceedings of the 2019 6th International Conference on Computing for Sustainable Global Development, INDIACom 2019*, 475–480.

[48] Pestana, J. (n.d.). *Smartbox: Caixa de correio inteligente*.

[49] Chaves, F., & Mendes, L. (n.d.). *Home Assistant para Gestão de Casas Inteligentes*.

[50] Franco, J., Gouveia, J., & Silva, J. (n.d.). *Gestão de Tomadas Inteligentes*.

[51] *Hubitat Elevation™ | Local, Reliable, Fast and Private Home Automation – HUBITAT*. (n.d.). Retrieved January 17, 2021, from https://hubitat.com/

[52] *Smart Home - Home Monitoring, SmartThings | Samsung US*. (n.d.). Retrieved September 2, 2021, from https://www.samsung.com/us/smartthings/

[53] *Samsung SmartThings App: What It Is and How to Use It*. (n.d.). Retrieved January 9, 2022, from https://www.lifewire.com/what-is-the-samsung-smartthings-app-5104981

[54] *Hubitat vs SmartThings - Home Automation*. (n.d.). Retrieved June 18, 2022, from https://home-automations.net/smartthings-vs-hubitat/

[55] *Compare Hubitat Vs SmartThings - Which One's Better? - DIY Smart Home Hub*. (n.d.). Retrieved June 18, 2022, from https://www.diysmarthomehub.com/hubitat-vs-smartthings/

[56] Helal, S., & Chen, C. (2009). *The Gator Tech Smart House*. 1. https://doi.org/10.1145/1592700.1592715

[57] Sanchez, V. G., Pfeiffer, C. F., & Skeie, N. O. (2017). A review of smart house analysis methods for assisting older people living alone. *Journal of Sensor and Actuator Networks*, *6*(3), 1–38. https://doi.org/10.3390/jsan6030011

[58] *Smart lighting | Philips Hue*. (n.d.). Retrieved September 1, 2021, from https://www.philips-hue.com/en-us

[59] *iOS - Home - Apple*. (n.d.). Retrieved September 2, 2021, from https://www.apple.com/ios/home/

[60] *TRON Intelligent House*. (n.d.). Retrieved August 30, 2021, from http://tronweb.super-nova.co.jp/tronintlhouse.html

[61] *House of Controversy: Technology: A new computer standard automates the TRON home with 1,000 microprocessors. Some American firms, however, are leery of Japanese intentions. - Los Angeles Times*. (n.d.). Retrieved January 28, 2021, from https://www.latimes.com/archives/la-xpm-1990-07-02-fi-598-story.html

[62] Cook, D. J., Youngblood, M., Heierman, E. O., Gopalratnam, K., Rao, S., Litvin, A., & Khawaja, F. (2003). MavHome: An agent-based smart home. *Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications, PerCom 2003*, 521–524. https://doi.org/10.1109/percom.2003.1192783

[63] Sanchez, V. G., Pfeiffer, C. F., & Skeie, N. O. (2017). A review of smart house analysis methods for assisting older people living alone. *Journal of Sensor and Actuator Networks*, *6*(3), 1–38. https://doi.org/10.3390/jsan6030011

[64] Weiser, M. (1999). The computer for the 21st century. *ACM SIGMOBILE Mobile Computing and Communications Review*, *3*(3), 3–11. https://doi.org/10.1145/329124.329126

[65] Aarts, E., & De Ruyter, B. (2009). New research perspectives on ambient intelligence. *Journal of Ambient Intelligence and Smart Environments*, *1*(1), 5–14. https://doi.org/10.3233/AIS-2009-0001

[66] Minker, W., López-Cózar, R., & McTear, M. (2009). The role of spoken language dialogue interaction in intelligent environments. *Journal of Ambient Intelligence and Smart Environments*, *1*(1), 31–36. https://doi.org/10.3233/AIS-2009-0004

[67] Ramos, C., Augusto, J. C., & Shapiro, D. (2008). Ambient intelligence the next step for artificial intelligence. *IEEE Intelligent Systems*, *23*(2), 15–18. https://doi.org/10.1109/MIS.2008.19

[68] B.J. Fogg. 2003. Persuasive Technology: Using Computers to Change What We Think and Do. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[69] Tulenkov, A., Parkhomenko, A., Sokolyanskii, A., Stepanenko, A., & Zalyubovskiy, Y. (2018). The features of wireless technologies application for smart house systems. *Proceedings of the 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems, IDAACS-SWS 2018*, 1–5. https://doi.org/10.1109/IDAACS-SWS.2018.8525842

[70] Nitin Naik. (2017). Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP Nitin. *2017 IEEE International Systems Engineering Symposium (ISSE)*. https://doi.org/10.1109/SysEng.2017.8088251

[71] Yokotani, T., & Sasaki, Y. (2017). Comparison with HTTP and MQTT on required network resources for IoT. *ICCEREC 2016 - International Conference on Control, Electronics, Renewable Energy, and Communications 2016, Conference Proceedings*, 1–6. https://doi.org/10.1109/ICCEREC.2016.7814989

[72] *Why HTTP is not suitable for IOT applications. - Concurrency*. (n.d.). Retrieved December 6, 2020, from https://www.concurrency.com/blog/june-2019/why-http-is-not-suitable-for-iot-applications

[73] Wukkadada, B., Wankhede, K., Nambiar, R., & Nair, A. (2018). Comparison with HTTP and MQTT in Internet of Things (IoT). *Proceedings of the International Conference on Inventive Research in Computing Applications, ICIRCA 2018*, *Icirca*, 249–253. https://doi.org/10.1109/ICIRCA.2018.8597401

[74] *MQTT - The Standard for IoT Messaging*. (n.d.). Retrieved December 6, 2020, from https://mqtt.org/

[75] *Understanding MQTT Topics*. (n.d.). Retrieved March 20, 2021, from http://www.steves-internet-guide.com/understanding-mqtt-topics/

[76] Sikandar, M., Khiyal, H., Khan, A., Shehzadi, E., Kodali, R. K., Soratkal, S., Conference, G., Technologies, C., & Dfcg, E. J. (2015). *MQTT based Home Automation System Using ESP8266*. *365565*(Gcct), 807–813.

[77] Raghunathan, S., Prasad, A., Mishra, B. K., & Chang, H. (2005). Open source versus closed source: Software quality in monopoly and competitive markets. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, *35*(6), 903–918. https://doi.org/10.1109/TSMCA.2005.853493

[78] Raymond, E. S. (2008). *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. http://books.google.com/books?id=F6qgFtLwpJgC&pgis=1

[79] *What Is Windows 10 IoT, and When Would You Want to Use It?* (n.d.). Retrieved October 18, 2021, from https://www.howtogeek.com/413102/what-is-windows-10-iot-and-when-would-you-want-to-use-it/

[80] Liming, B. S. D., & Malin, J. R. (2015). *Windows 10 IoT – The Big Reboot*. 1–7.

[81] *Evolution of Windows IoT: The foundation for your intelligent edge | Windows Experience Blog*. (n.d.). Retrieved November 1, 2021, from https://blogs.windows.com/windowsexperience/2019/04/03/evolution-of-windows-iot-the-foundation-for-your-intelligent-edge/

[82] *ESP8266 - Wikipedia*. (n.d.). Retrieved June 1, 2022, from https://en.wikipedia.org/wiki/ESP8266

[83] *ESPHome — ESPHome*. (n.d.). Retrieved June 1, 2022, from https://esphome.io/

[84] *SONOFF Zigbee 3.0 USB Dongle Plus - SONOFF Official*. (n.d.). Retrieved June 1, 2022, from https://sonoff.tech/product/diy-smart-switch/sonoff-dongle-plus/

[85] *CC2531 data sheet, product information and support | TI.com*. (n.d.). Retrieved June 1, 2022, from https://www.ti.com/product/CC2531

[86] *Smart lighting | Philips Hue*. (n.d.). Retrieved September 1, 2021, from https://www.philips-hue.com/en-us

[87] *Philips Hue: Supported lights and devices [Hue compatible] (Hue: Unterstützte Lampen & Geräte [Hue-kompatibel]) | iConnectHue*. (n.d.). Retrieved July 12, 2022, from https://iconnecthue.com/supported-devices/

[87] *Home Assistant Developer Docs | Home Assistant Developer Docs*. (n.d.). Retrieved June 2, 2021, from https://developers.home-assistant.io/

[88] *[On hold] Deprecating Home Assistant Supervised on generic Linux - Home Assistant*. (n.d.). Retrieved December 29, 2020, from https://www.home-assistant.io/blog/2020/05/09/deprecating-home-assistant-supervised-on-generic-linux/

[89] home-*assistant/supervised-installer: Installer for a generic Linux system*. (n.d.). Retrieved December 29, 2020, from https://github.com/home-assistant/supervised-installer

[90] *Buildroot - Making Embedded Linux Easy*. (n.d.). Retrieved June 4, 2022, from https://buildroot.org/

[91] *Atualizações de sistema A / B (contínuas) | Android Open Source Project*. (n.d.). Retrieved June 4, 2022, from https://source.android.com/devices/tech/ota/ab

[92] *Hubitat Elevation Documentation - Hubitat Documentation*. (n.d.). Retrieved June 5, 2022, from https://docs.hubitat.com/index.php?title=Hubitat_Elevation_Documentation

[93] *Programming Reference - Get Help - Hubitat*. (n.d.). Retrieved July 26, 2022, from https://community.hubitat.com/t/programming-reference/88884/3

[94] *GitHub - hubitat/HubitatPublic*. (n.d.). Retrieved July 26, 2022, from https://github.com/hubitat/HubitatPublic

[95] *Hubitat native drivers - closed or open source? - Developers - Hubitat*. (n.d.). Retrieved July 26, 2022, from https://community.hubitat.com/t/hubitat-native-drivers-closed-or-open-source/11380

[96] AlexxIT. (n.d.). *Home Assistant Windows Portable (HassWP) - GitHub*. https://github.com/AlexxIT/HassWP

[97] *Home Assistant vs. Home Assistant Core - Home Assistant*. (n.d.). Retrieved February 13, 2022, from https://www.home-assistant.io/faq/ha-vs-hassio/

[98] *Database of Zigbee devices compatible with ZHA, Tasmota, Zigbee2MQTT, deCONZ, ZiGate and ioBroker*. (n.d.). Retrieved July 3, 2022, from https://zigbee.blakadder.com/

[99] *Home | Zigbee2MQTT*. (n.d.). Retrieved July 5, 2022, from https://www.zigbee2mqtt.io/

[100] *Hubitat Vs Home Assistant - Home Automation*. (n.d.). Retrieved January 17, 2021, from https://home-automations.net/hubitat-vs-home-assistant/