UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE
DEPARTAMENTO DE COMPUTAÇÃO

RENAN FERREIRA MUNIZ

AUTOMAÇÃO RESIDENCIAL BASEADA EM PLATAFORMAS DE CÓDIGO ABERTO COM BAIXO CUSTO

ALEGRE
2021

RENAN FERREIRA MUNIZ

AUTOMAÇÃO RESIDENCIAL BASEADA EM PLATAFORMAS DE CÓDIGO ABERTO COM BAIXO CUSTO

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Helder de Amorim Mendes.

ALEGRE

2021

RENAN FERREIRA MUNIZ

AUTOMAÇÃO RESIDENCIAL BASEADA EM PLATAFORMAS DE CÓDIGO ABERTO COM BAIXO CUSTO

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Aprovado em ____ de maio de 2021.

COMISSÃO EXAMINADORA

_____

Prof. Dr. Helder de Amorim Mendes
Universidade Federal do Espírito Santo
Orientador

_____

Prof. Dr. Clayton Vieira Fraga Filho
Universidade Federal do Espírito Santo

_____

Prof. Dr. Antonio Almeida de Barros Junior
Universidade Federal do Espírito Santo

RESUMO

Ao longo da história, a tecnologia busca sempre transformar a forma que interagimos com o ambiente em algo mais simples e muitas vezes de forma automática. O mundo da automação residencial não é diferente, com o crescimento da Internet das Coisas, o acesso à dispositivos inteligentes se tornou acessível e a possibilidade de sistemas de automação residencial de baixo custo é cada vez mais viável a todos. Com esse intuito, o presente trabalho será apresentado um projeto de sistema de automação residencial de baixo custo baseado em plataformas de código aberto, com o objetivo de automatizar uma casa sem a necessidade de modificar estruturalmente o ambiente, desenvolvendo módulos com conexão sem fio e uma plataforma central de controle para prover ao usuário o envio de comandos e recebimento de informações do estado do ambiente. O desenvolvimento e construção dos módulos de controle e da central foram bem-sucedidos, permitindo o envio de comandos e recebimento dos estados de cada dispositivo individualmente, através de uma interface acessada por navegador *desktop* ou aplicativo *mobile*. O custo final do projeto esteve dentro de um limite considerado baixo, portanto, resultando em uma alternativa viável a muitas pessoas que pretendem realizar esse pequeno investimento em uma solução de automação residencial.

Palavras chave: Automação residencial, Casa inteligente*, Smart Home*, Internet das Coisas.

ABSTRACT

Throughout the history, the technology always seeks to transform into a simpler way we interact with the environment, and often into an automatic way. The home automation area is not different, with the growth of the Internet of Things, the access to smart devices became acessible and the possibility to have a low cost home automation system is increasingly viable for everyone. For that purpose, this work will be presented a project of a low cost home automation system based on open source platforms, with the objective to automate a house without change structurally the environment, developing wireless connection modules and a central control platform providing to the user a way to send commands and receive information of the current environment state. The development and construction of the control modules and the central platform were successful, allowing to send commands and receive individually the states of each device, through an user interface accessed via desktop browser or mobile application. The project budget was within a limit considered low, so the result obtained was a viable alternative for the many people who intend to make this small investment in a home automation solution.

Keywords: Home Automation, Smart Home, Internet of Things, Smart Devices.

LISTA DE FIGURAS

Figura 1- Arquitetura Sure Universal de casa inteligente 15

Figura 2 - Dispositivos e empresas fabricantes.....16

Figura 3 - Chip ESP8266 utilizado no projeto 17

Figura 4 - Modelo Sonoff Basic 19

Figura 5 - Fluxo do protocolo MQTT 21

Figura 6 - Apresentação hierárquica dos conceitos openHAB 26

Figura 7 - Modelos de interruptores Sonoff TX.....30

Figura 8 - Sintaxe de uma regra escrita 31

Figura 9 - Arquitetura do sistema desenvolvido 33

Figura 10 - Resultado do módulo sensor para portas ou janelas 34

Figura 11 - Página de configuração do dispositivo.....34

Figura 12 - Interruptor Sonoff TX de duas teclas instalado 35

Figura 13 - Página de configuração do módulo interruptor de iluminação 35

Figura 14 - Módulo de controle da fita RGB concluído.....36

Figura 15 - Módulo RGB instalado com sensor de temperatura adaptado.....36

Figura 16 - Regra para criação de cena modo filme 37

Figura 17 - Automatização das luzes ao reproduzir um vídeo 38

Figura 18 - Automatização de tarefas caso a porta mude de estado 38

Figura 19 - Interface do openHAB acessada por navegador 40

Figura 20 - Interface mobile do aplicativo openHAB 41

LISTA DE TABELAS

Tabela 1 - Conceitos e significados do openHAB 25

Tabela 2 - Custo total do projeto 41

LISTA DE ABREVIATURAS E SIGLAS

GPIO - *General Purpose Input Output*

IDE - *Integrated Development Environment*

IoT - *Internet of Things* ou Internet das Coisas

IP - *Internet Protocol* ou Endereço IP

LED - *Light Emitting Diode* ou Diodo Emissor de Luz

MQTT - *Message Queuing Telemetry Transport*

openHAB - *open Home Automation Bus*

RGB - *Red, Green, Blue* ou Vermelho, Verde, Azul

SD - *Secure Digital*

Wi-Fi - *Wireless Fidelity*

SUMÁRIO

1 INTRODUÇÃO 11
1.1 O PROBLEMA E SUA IMPORTÂNCIA 11
1.2 OBJETIVOS 12
1.2.1 Objetivo Geral 12
1.2.2 Objetivos Específicos 12
2 REFERENCIAL TEÓRICO 13
2.1 SMART HOME, OU A CASA INTELIGENTE 13
2.2 MOTIVOS PARA TORNAR A SUA CASA INTELIGENTE 14
2.3 A ARQUITETURA E COMPONENTES DE UMA CASA INTELIGENTE 15
2.3.1 Hardwares para configuração de casas inteligentes 16
2.3.2 Sensores para automação de casas inteligentes 17
2.3.3 Atuadores para casas inteligentes 19
2.3.4 Protocolos de comunicação 20
2.4 REVISÃO BIBLIOGRÁFICA 21
3 METODOLOGIA 24
3.1 SISTEMA CENTRAL DE CONTROLE 24
3.1.1 OpenHAB 24
3.1.1.1 Instalação 25
3.1.1.2 Conceitos 25
3.2 PLANEJAMENTO DOS MÓDULOS 27
3.2.1 Módulo sensor de porta ou janela 28
3.2.2 Módulo interruptor simples 28
3.2.3 Módulo interruptor de iluminação 29
3.2.4 Módulo LED RGB com sensor de temperatura 30
3.3 TORNANDO MAIS INTELIGENTE 31
4 RESULTADOS OBTIDOS 33
4.1 MÓDULOS 33

4.1.1 Módulo sensor de porta 33

4.1.2 Módulo interruptor simples para ventilador 34

4.1.3 Módulo interruptor de iluminação 35

4.1.4 Módulo LED RGB com sensor de temperatura 36

4.2 REGRAS CRIADAS PARA A AUTOMAÇÃO DAS TAREFAS 37

4.3 CENTRAL DE CONTROLE OPENHAB 39

4.4 CUSTO FINAL DO PROJETO 41

5 CONCLUSÕES 42

REFERÊNCIAS BIBLIOGRÁFICAS 43

APÊNDICES 46

APÊNDICE A – Código fonte do módulo sensor de porta 46

APÊNDICE B – Código fonte do módulo controlador fita LED RGB 50

APÊNDICE C – Arquivo de definição de Things: quarto.things 56

APÊNDICE D – Arquivo de definição de Items: quarto.items 57

APÊNDICE E – Arquivo de definição de Items: midia.items 58

APÊNDICE F – Arquivo de configuração do Sitemap: quarto.sitemap 59

APÊNDICE G – Arquivo de definição das Rules: quarto.rules 60

APÊNDICE H – Arquivo de definição das Rules: fitaled.rules 62

APÊNDICE I – Arquivo de transformação: cena.map 63

APÊNDICE J – Arquivo de transformação: lua.map 64

APÊNDICE K – Arquivo de transformação: porta.map 65

1 INTRODUÇÃO

1.1 O PROBLEMA E SUA IMPORTÂNCIA

Atualmente com a necessidade de o mundo estar totalmente conectado e automatizado, cresce o interesse das pessoas em automação residencial por trazer comodidade e facilidade ao realizar tarefas do cotidiano.

A partir dessa necessidade, a Internet das Coisas ou *Internet of Things* (IoT), surge como uma evolução da internet atual que promete revolucionar os meios de como interagimos na sociedade e no próprio meio ambiente, através de equipamentos físicos ou virtuais, em que os limites entre eles estarão cada vez menores (LACERDA; LIMA-MARQUES, 2015).

De acordo com Lueth (2018), no ano de sua pesquisa o número de dispositivos de *IoT* (Internet of Things) conectados no mundo eram de 7 bilhões, e que esse número de dispositivos de que estão ativos deve crescer para 10 bilhões até 2020 e 22 bilhões até 2025.

Robles e Kim (2010) afirmaram que “O campo de automação residencial está se expandindo rapidamente à medida que as tecnologias eletrônicas convergem. A rede doméstica abrange comunicações, entretenimento, segurança, conveniência e sistemas de informação”, e concluem que os ambientes inteligentes para apoiar idosos e deficientes possui um enorme potencial, através de uma casa inteligente com a integração da tecnologia e serviços pela da rede para se obter uma melhor qualidade de vida.

O objetivo de um sistema de automação é que o *software* de que desempenha o papel de um agente inteligente perceba o estado do ambiente físico e os moradores usando sensores, em seguida, realiza ações para alcançar objetivos específicos, como maximizar conforto das pessoas, minimizando o consumo de recursos e mantendo a saúde e a segurança do lar e dos moradores (COOK, 2012).

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Este trabalho tem como objetivo propor um projeto de automação residencial visando ter um orçamento de baixo custo para a implantação através de recursos e ferramentas livres disponibilizadas gratuitamente.

Apresentar uma alternativa de automação residencial de fácil acesso, e que com poucos recursos é possível criar um sistema simples, flexível e eficiente.

1.2.2 Objetivos Específicos

Projetar e implementar módulos para controle e monitoramento dos dispositivos individualmente, utilizando-se de sensores e atuadores adequados para cada situação em que for necessário.

Projetar e implementar uma central de controle para dispositivos com os mais variados protocolos de comunicação de forma automática.

Realizar testes práticos para a validação do sistema, funcionamento e integração entre os módulos e comunicação dos protocolos utilizados.

2 REFERENCIAL TEÓRICO

2.1 SMART HOME, OU A CASA INTELIGENTE

Inicialmente, quando falamos em casa inteligente pode soar um pouco vago o que “inteligente” significa, para isso, precisamos saber o que são esses conceitos de casa inteligente e como funcionam.

De acordo com Miller (2017) uma casa que é chamada de inteligente usa a tecnologia capaz de automatizar e operar dispositivos e tarefas importantes que são: Iluminação, temperatura, portas e campainhas, sistema de segurança, entretenimento e eletrodomésticos. Além disso, ele afirma que: “Tecnologia inteligente é a tecnologia que sente o que está acontecendo ao redor de um sensor ou dispositivo em particular e age de forma autônoma baseada nas informações que foram coletadas”. Fabi, Spigliantini e Corgnati (2016) complementam esse conceito: “Apenas nos últimos tempos, o termo “Smart” ou “Inteligente” está relacionado à eficiência energética em edifícios. Assim, muitas vezes se utiliza de um sistema de gerenciamento de energia, monitorando o consumo dentro das residências. ”.

Outro ponto fundamental de um sistema de automação residencial se passa pela comunicação com seus componentes. Para a comunicação é possível ter uma rede cabeada ou sem fio, e a escolha da mais adequada se passa muito pela infraestrutura local e o orçamento disponível. A opção por uma rede cabeada é a mais confiável, entretanto, pode se tornar inviável a passagem de cabos, e por conta disso, os projetos de casa inteligente costumam utilizar muito das redes sem fio que cada vez mais se mostram confiáveis para a transmissão dos dados.

O conceito apresentado de casa inteligente faz parte de um grupo maior chamado Internet das Coisas (*IoT*) em que Miller (2017) define como: “Na IoT, vários dispositivos inteligentes conectam entre si, dentro e fora da casa, para permitir uma operação automatizada e de tomada de decisão”. Uma vez que estes dispositivos estão conectados à internet, se torna possível controlá-los de qualquer local do mundo.

2.2 MOTIVOS PARA TORNAR A SUA CASA INTELIGENTE

O uso de um sistema de automação residencial se baseia em controlar seus dispositivos através de um aplicativo móvel, aplicação web ou até mesmo por um alto falante inteligente que recebe comandos de voz e executa o que foi ordenado. Há alguns anos atrás isso poderia parecer distante, mas hoje em dia já é a realidade e não exige um grande investimento para tal (ROBLES e KIM, 2010).

A comodidade de acender uma lâmpada, controlar o sistema de som, ligar um ar condicionado na palma da mão, e tudo isso feito em um único dispositivo é algo bastante tentador. Não só apenas controlar os dispositivos, mas deixá-los configurados para quando estiver escurecendo as lâmpadas do quintal acenderem, quando um filme estiver exibindo na sua televisão, a iluminação é reduzida para um modo mais confortável, ou quando começar a chover receberá um alerta no seu celular que determinada janela está aberta, se alguém tocar a campainha, no celular poderá ver quem está na sua porta. Essas são algumas das praticamente infinitas soluções em que um sistema de automação residencial entra em ação, mostrando que atualmente não existem limites para a casa inteligente (MILLER, 2017).

A comodidade não é o único fator, a economia de tempo e esforço também entra em questão. Levantar e acender uma lâmpada não é algo difícil de se fazer, entretanto ninguém se vê levantando do sofá apenas para trocar de canal da televisão ou só para aumentar o volume. Portanto, são coisas que somadas resultam em alguns segundos que poderiam estar sendo direcionados para outra coisa em que seu tempo vale mais.

Economia de energia também faz parte do pacote de uma casa inteligente, quando não tiver nenhuma pessoa em casa o sistema poderá ser configurado para desligar alguma lâmpada que estiver acesa ou algum outro aparelho eletrônico, o monitoramento em tempo real de quanto está sendo gasto com cada tipo aparelho ou determinado cômodo da casa poderá facilitar o entendimento do consumo de energia de uma residência (FABI, SPIGLIANTINI e CORGNATI, 2016).

Para a maioria das pessoas um sistema de automação é considerado algo supérfluo e que não precisam de um, mas em casos de pessoas com deficiência física ou com problemas de locomoção, um sistema de automação residencial proporciona uma facilidade muito grande para realizarem tarefas simples, mas que com as limitações se tornam algo desgastante e muito difícil, sendo assim fundamental para que tenham uma independência dentro de suas próprias casas (WAHLBERG, 2018).

2.3 A ARQUITETURA E COMPONENTES DE UMA CASA INTELIGENTE

Atualmente, diversas empresas possuem a sua arquitetura de casa inteligente com seus próprios dispositivos desenvolvidos, tornando em um ecossistema integrado. A maioria delas seguem um padrão que é composto de um dispositivo chamado de *gateway* sendo responsável por conectar os diversos equipamentos (lâmpadas, interruptores, sensores e etc.) a uma rede local, e que este *gateway* é conectado à uma aplicação na nuvem que irá prover o controle e acesso através de um aplicativo móvel proprietário. A Figura 1 ilustra esse modelo:

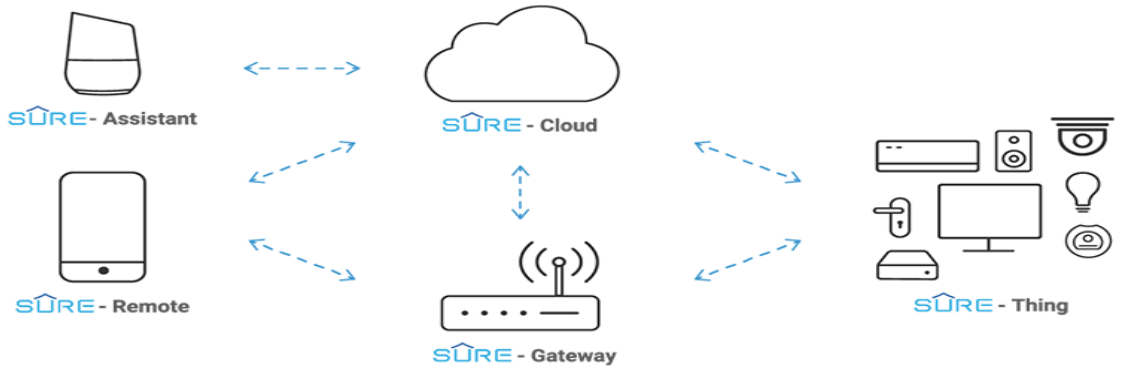

Figura 1- Arquitetura Sure Universal de casa inteligente
Fonte: Sure Universal, 2019

Da mesma forma que algumas empresas possuem o seu próprio ecossistema, outras preferem comercializar apenas os dispositivos e liberá-los para quem desejar fazer uma integração ao seu sistema. Companhias como Google e Amazon possuem seus *Hubs* que são assistentes de voz e capazes de controlar esses dispositivos. Na Figura 2, é apresentado esse conjunto de marcas que se integram aos assistentes Google Home e Amazon Alexa:

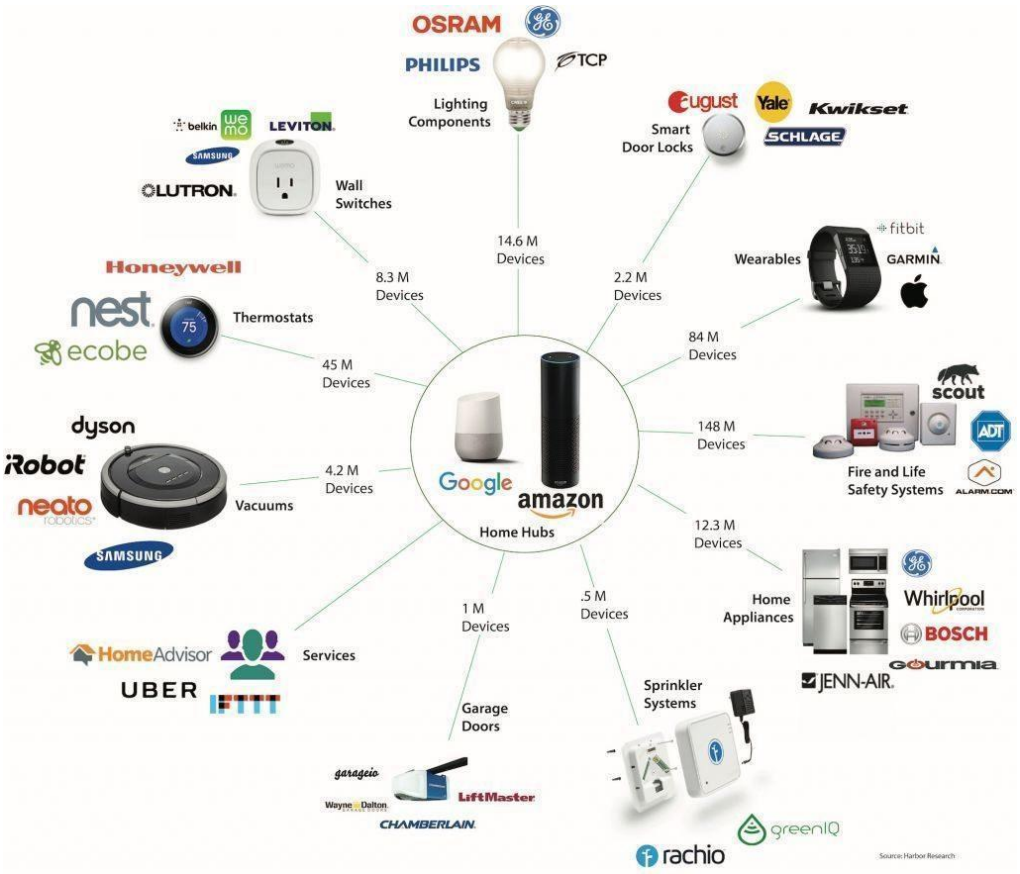

Figura 2 - Dispositivos e empresas fabricantes
Fonte: Harbor Research, 2017

2.3.1 Hardwares para configuração de casas inteligentes

Entusiastas e estudantes de *IoT* que produzem pequenos projetos de automação, normalmente utilizam *hardwares* ou placas simples de desenvolvimento que possuem a capacidade de serem programadas, tornando possível obter dados de sensores e controlar atuadores sem a necessidade de construírem elaboradas placas de circuito impresso. As mais populares no mundo atualmente são: Arduino, Raspberry PI e as placas baseadas nos chips da família ESP.

Arduino é uma plataforma eletrônica de código aberto com o objetivo de ser fácil de utilizar o *software* e *hardware*, tendo um baixo custo. A orgazinação Arduino afirma que as placas Arduino vem sendo utilizadas por milhares de diferentes tipos de projeto e aplicações, o *software* é fácil de utilizar para os iniciantes e ao mesmo tempo é flexível o suficiente para os usuários avançados. Existem diversos tipos e modelos de placa, cada um com um propósito e uso adequado para cada situação ou objetivo que deseja no projeto.

A Raspberry PI é uma fundação baseada no Reino Unido que produz computadores de tamanho reduzido, baixo custo e alta performance para pessoas que desejam aprender programação, resolver problemas ou se divertirem. As placas Raspberry PI possuem um poder computacional muito maior comparado aos Arduinos, e isso deu a capacidade de rodar um sistema operacional baseado no *Debian* chamado de *Raspbian,* como possui um processamento de um computador simples, geralmente é utilizada para funções mais elaboradas como: Servidores Web, servidor de mídia, central de controle e automação e entre outras inúmeras aplicações.

Uma solução também muito utilizada nos projetos de *IoT* são os chips da família ESP. A fabricante Espressif Systems desenvolve diversos módulos com os micros controladores, dentre os mais famosos ESP8266 e ESP32 que ambos possuem conexão à rede por WiFi, com a diferença do segundo possuir *bluetooth* e mais capacidade de processamento como as principais características.

Figura 3 - Chip ESP8266 utilizado no projeto
Fonte: Elaborado pelo autor

Todos os *hardwares* citados acima possuem portas de entrada e saída, que permitem serem programadas para receberem sinais de um sensor (Entrada) ou emitirem comandos (Saída) para um atuador ou qualquer dispositivo capaz de realizar alguma ação no ambiente.

2.3.2 Sensores para automação de casas inteligentes

Como todo sistema de automação é necessário perceber o ambiente ao seu redor, os sensores são os encarregados dessa tarefa, (MILLER, 2017).

Começando com os sensores de temperatura, que permitem fazer a leitura da temperatura do ambiente. Um exemplo de sensor muito utilizado nos projetos é o sensor DHT11 que é produzido pela empresa Aosong Eletronic, este sensor realiza leituras de temperaturas entre -40ºC e 80ºC e umidade de 0% a 100%.

Sensores de luminosidade são também muito utilizados, por terem a sensibilidade de perceber a luz e informar ao sistema quando algum ambiente estiver iluminado ou apagado.

Um dos sensores mais importantes em um sistema de automação residencial é o de porta e janela, que é um detector magnético com duas partes que quando em contato, um circuito elétrico é fechado e assim a informação é recebida pelo módulo responsável pelo sensor.

Outro sensor com grande utilidade é o de presença e movimento, que usa sensores infravermelhos para detectar a movimentação de pessoas e objetos, quando o mesmo é acionado, um sinal elétrico é emitido para o respectivo módulo que irá decidir qual ação será tomada. Este sensor é comumente utilizado no controle de iluminação e também para detectar se possui alguma pessoa em determinado ambiente.

Para identificar o consumo de energia de determinado aparelho eletrônico, de um cômodo ou até mesmo de uma casa inteira, geralmente são usados os sensores de corrente. Estes sensores medem a corrente elétrica presente em um cabo de energia e enviam para a placa controladora o valor medido, e que lá é processado a informação e capaz de informar quanto se está consumindo no exato momento.

A segurança dos moradores presentes em uma residência é algo fundamental, com o intuito de proteger, os sensores de gás e fumaça são grandes aliados para essa aplicação. Existem sensores que conseguem medir diversos tipos de componentes inflamáveis presentes no ar com apenas um módulo sensorial, como por exemplo o Sensor de Gás MQ-2, quando os parâmetros detectados se apresentam acima dos valores ideais, é disparado um sinal eletrônico pelas portas de entrada e saída de um Arduino ou ESP8266 que irá processar e enviar a mensagem para onde foi programado.

Outro sensor de funcionamento simples e que não é muito utilizado nos projetos, mas possui uma aplicação interessante é o sensor de chuva e líquidos, que consiste em uma placa de circuito eletrônico exposta para detectar a presença de chuva ou algum outro líquido, enviando os dados para um microcontrolador encarregado do sensor.

Existem outros sensores que também podem ser incorporados a um sistema de automação, que são: Acelerômetro e giroscópio, sensor de obstáculos infravermelho, sensor de fluxo de água, sensor de chama e fogo, pressão barométrica, sensor de nível de água, sensor de som e sensor de umidade do solo são alguns que também poderiam fazer parte de um sistema de automação *IoT*, ficando de acordo com a necessidade de ter ou não tais informações coletadas por eles.

2.3.3 Atuadores para casas inteligentes

O universo dos atuadores no sistema de automação é bem menor comparado aos sensores, e um dos principais, se não o principal atuador é o Relé, que consiste em ser um interruptor eletromecânico que quando recebe um sinal elétrico, a passagem de energia é liberada ou interrompida de acordo com o sistema de ligação que foi feito. Este atuador está presente em diversos módulos inteligentes de automação, dentre eles: Interruptor de iluminação, tomada elétrica, portão eletrônico e interruptores em geral são os mais comuns de se encontrar esse dispositivo. A fabricante Songle possui os relés mais utilizados nos projetos, sendo incorporados em vários módulos com diferentes tipos de tensão elétrica.

A fabricante chinesa ITEAD possui como um de seus produtos, o atuador Sonoff Basic, esse dispositivo tem capacidade de ser um interruptor que conecta à rede *wireless* para um aparelho eletrônico com consumo de até 2200 *Watts* de potência, é baseado no chip ESP8266 e utiliza-se de relés para realizar a ação de interruptor elétrico, que é comandado através da rede.

Figura 4 - Modelo Sonoff Basic
Fonte: ITEAD INTELLIGENT SYSTEMS, 2021

Outro atuador amplamente utilizado é o emissor de infravermelho, capaz de controlar dispositivos que utilizam da tecnologia, como por exemplo: Televisão, rádio, receptor, ventilador e aparelhos de multimídia em geral.

Uma forma de o sistema atuar no ambiente também se dá pelo uso de válvulas solenoides que quando alimentadas por corrente elétrica, permite a vazão do fluxo de água, e corta o fluxo quando deixa de ser energizada. Podendo ser aplicada para sistemas de irrigação automática ou para controles de reservatórios de água.

Por meio de motores o sistema também é capaz de efetuar ações no ambiente, sendo assim possível controlar um portão de garagem ou cortinas por exemplo através desse atuador.

2.3.4 Protocolos de comunicação

Os protocolos de comunicação podem ser definidos como: "Conjunto de regras, padrões e especificações técnicas que regulam a transmissão de dados entre computadores por meio de programas específicos, permitindo a detecção e correção de erros; protocolo de transmissão de dados" (PASTORE, 2003).

Pastore (2003) atribui uma vantagem aos protocolos de comunicação, que independem do sistema de cada dispositivo em que está sendo implementado, já que possuem regras que padronizam e uniformizam a comunicação, permitindo que todos falem a mesma “língua”.

Para o mundo da automação residencial, as grandes fabricantes geralmente utilizam os protocolos *Z-Wave* e *Zigbee*. Kaven (2005) definiu *“*A tecnologia Z-Wave é um protocolo de rede em malha bidirecional baseado em radiofrequência”, e que “*é* o recurso de rede em malha que permite que os sinais sem fio sejam roteados de dispositivo para dispositivo até que o dispositivo de destino seja atingido. ”

O protocolo *Zigbee* foi explicado por Frenzel (2012) como sendo muito parecido no funcionamento com *Z-Wave* mas atuando numa frequência mais alta que diminui o seu alcance*.* O *Zigbee* possui uma maior flexibilidade, por outro lado, o Z-Wave é mais simples ao incorporá-lo em um projeto por ser menos complexo o seu desenvolvimento.

Um dos principais protocolos no mundo *IoT* é o protocolo MQTT. Yuan (2017) explica que o MQTT é um dos melhores protocolos de *IoT* existentes, por ser leve e flexível permite que seja utilizado em redes com uma velocidade baixa e

com alta latência. E o principal recurso deste protocolo é o modelo de publicação e assinatura (*Publish-Subscribe*), na rede existem dois tipos de entidade: *Broker* e os clientes. O *Broker* é um servidor que recebe mensagens dos clientes e encaminha para os outros clientes com interesse nessas mensagens. Já o cliente é qualquer um que possa interagir com o *broker* e receber mensagens, podendo ser um sensor de presença ou um interruptor. O funcionamento da comunicação se dá por um cliente fazendo a inscrição (*Subscribe*) em qualquer tópico, e podendo publicar (*Publish*) uma mensagem em qualquer tópico, que envia a mensagem e o tópico para o *broker* que encaminha para todos que possuem a inscrição neste tópico. A Figura 5 representa de forma simples como esse fluxo do protocolo segue:

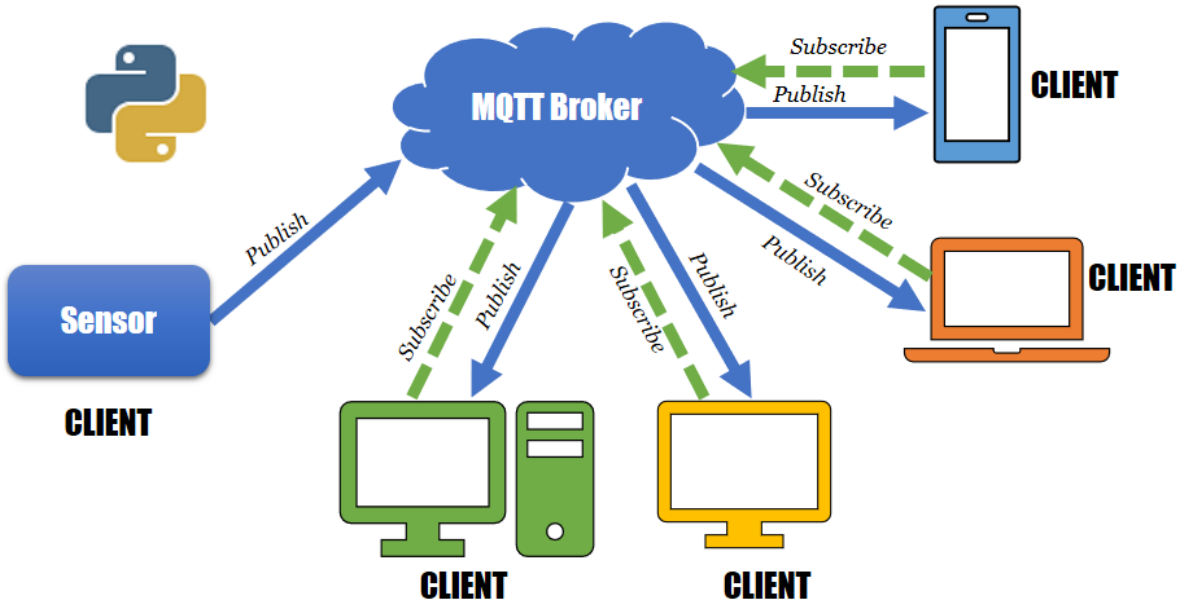

Figura 5 - Fluxo do protocolo MQTT
Fonte: Embedded Laboratory, 2018

2.4 REVISÃO BIBLIOGRÁFICA

Atualmente existem muitas empresas com soluções para automação residencial, entretanto o custo elevado e a necessidade de realizar modificações na estrutura da casa acabam se tornando inviáveis para a maioria das pessoas.

Em 2013 a empresa Optiart Solutions afirmou sobre o custo médio de um projeto de automação: “O custo de um projeto de automação de ambientes varia entre R$ 10 mil e R$ 30 mil. É o cliente que escolhe quais serviços terá à disposição, se eles serão integrados e como serão controlados à distância. ” Em entrevista ao (G1.GLOBO.COM) (2013).

Segundo a Associação Brasileira de Automação Residencial e Predial (Aureside) em 2017 afirmou de acordo com O Tempo (2017) que um projeto básico de automatização residencial custava a partir de R$ 6.000,00 em um imóvel de 70m², com controle de quatro aparelhos de mídia, dois condicionadores de ar e iluminação.

Também existem outras empresas que apenas comercializam os produtos e o cliente não precisa arcar com os custos de instalação. A Philips conta com um kit de iluminação chamado de Philips Hue que possui duas lâmpadas dimerizáveis e um interruptor, podendo ser controladas pelo aplicativo da empresa. A Xiaomi possui uma linha de automação chamada de Aqara, com sensores de porta e janela, interruptor sem fio, sensor de presença, sensor de temperatura, câmera de monitoramento, tomada *smart* e lâmpadas controláveis. Para conectar todos os dispositivos é necessário um gateway que faz toda a tradução dos comandos enviados do aplicativo para a rede *ZigBee* dos dispositivos.

Piton (2017) propôs um sistema de automação residencial baseado na plataforma em nuvem da IBM chamada de Bluemix, o projeto tem como principal característica a exigência que os dispositivos tenham conexão com a internet para que as informações que o cliente deseja sejam recebidas na plataforma de acesso.

Com outra perspectiva, a proposta deste trabalho visa isolar o sistema de automação da internet ao máximo, tornando independente da rede mundial de computadores, e tendo como benefícios o tempo de resposta entre os comandos dos dispositivos é quase imediata, a segurança se torna maior por estar em uma rede local e não é necessário um serviço externo ao que já se encontra na central para a aplicação funcionar.

Um projeto mais próximo do que este trabalho busca foi apresentado por Deval (2015), que propõe também um sistema de automação residencial com baixo custo e com ferramentas de código aberto, o foco do trabalho é uma aplicação mobile que envia os comandos para os dispositivos conectados à rede, que são: Dispositivo para controle de iluminação, televisão, portão e acesso ao sistema de câmeras de segurança.

Apesar de ser de baixo custo, o projeto anterior apresentou se de poucas soluções que uma automação residencial exige, como por exemplo, ter um sistema em que possa realizar ações de forma automática sem a necessidade de o usuário realizar um comando específico. A não utilização de sensores é sentida no sistema, que se utiliza de poucos dispositivos.

O projeto proposto neste trabalho é ter uma flexibilização maior entre os módulos de controle, um custo de implantação baixo e não depender de internet para o sistema funcionar. Como os módulos serão construídos baseados em plataformas de código aberto, a liberdade para modificá-lo e melhorá-lo de acordo com a situação e necessidade do projeto é muito maior que em sistemas proprietários. Segurança e transparência dos dados que estão sendo transmitidos na rede é outro fator de grande importância neste trabalho.

3 METODOLOGIA

Ao desenvolver este projeto, a central de controle foi o primeiro passo, pois a partir dela pode-se entender melhor o funcionamento do sistema e como os módulos poderiam realizar a integração. A partir da central estabelecida, foram realizadas pesquisas de como obter ou construir tais módulos de controle, que foram anteriormente planejados como objetivo de uma automação residencial simples e de baixo custo priorizando a utilização de plataformas de código aberto.

3.1 SISTEMA CENTRAL DE CONTROLE

O ponto chave de um sistema de automação é a sua central de controle, então é preciso que a mesma tenha a capacidade de se comunicar com os módulos e ter um sistema operacional que possibilite o desenvolvimento e instalação das aplicações necessárias. A fim de reduzir custos, foi escolhida como o “cérebro” do sistema a placa Raspberry PI 3B, que é um computador simples, mas com dimensões, consumo e custo reduzidos.

O software como plataforma de automação central poderia ser instalado em um qualquer computador normal, porém estaria consumindo energia e espaço físico desnecessários em comparação à solução que foi escolhida. Como o sistema de automação que será usado no projeto não exige muito recurso computacional, a Raspberry PI 3B tem poder suficiente para essa tarefa.

3.1.1 OpenHAB

Para a central do sistema poder funcionar, é necessária uma plataforma que possua as soluções de integração entre diversos tipos de aplicações e protocolos, e que forneça uma interface amigável de acesso ao sistema e controle dos dispositivos. Atualmente, existem algumas plataformas de automação residencial de código aberto que realizam essa tarefa de ser o centro do sistema, como por exemplo: openHAB, Home Assistant e Domoticz são as mais populares. Neste projeto foi definido o openHAB como essa plataforma, que é desenvolvida em Java e se mostrou mais flexível quanto a conexão dos dispositivos e modificação das interfaces de usuário, além de serem mais intuitivas e agradáveis, tendo uma comunidade bastante ativa no desenvolvimento de novos recursos, suporte e atualizações frequentes.

A própria organização openHAB se define como "... uma plataforma de automação residencial de código aberto e independente da tecnologia, funciona como o centro da sua casa inteligente! ".

Um dos principais motivos de se ter uma central que executa um *software* de automação dentro de casa é eliminar a necessidade de conexão à internet para que o sistema continue em funcionamento, ao contrário dos modelos apresentados anteriormente, podendo funcionar perfeitamente *offline.*

3.1.1.1 Instalação

A instalação do openHAB na Raspberry PI é bem simples, ao entrar no site da plataforma[1] é apresentada a seção de *downloads*, onde poderá baixar o arquivo de imagem que será escrita em um cartão SD com auxílio do software Etcher. Concluído o processo de escrita no cartão, basta inserir na Raspberry, ligar à energia, conectar um cabo de rede e aguardar entre 15 e 45 minutos para o openHabian[2] executar a configuração inicial do sistema. Em seguida, acesse o endereço "http://openhabian:8080" na rede local para continuar o tutorial de primeiros passos da plataforma, (OPENHAB, 2021).

3.1.1.2 Conceitos

Aprofundando um pouco nos conceitos do openHAB, temos alguns que são os mais importantes e necessários para o funcionamento da plataforma:

Tabela 1 - Conceitos e significados do openHAB

Conceitos	**Significados**
Bindings	São componentes desenvolvidos por terceiros ou não, que provêm uma interface para interagir com dispositivos externos.
Things	"Coisas" que são apresentadas como a primeira representação dos dispositivos.
Channels	"Canais" que representam a conexão entre *Things* e *Items*
Items	"Itens" são a representação das informações geradas pelo openHAB sobre os dispositivos.

1 Site da plataforma openHAB: https://openhab.org.
2 Sistema operacional com openHAB integrado, feito para Raspberry PI baseado no Debian.

Rules	“Regras” que executam ações automáticas de acordo com um código escrito (“se isso acontecer, faça aquilo”).
Sitemap	É a interface do usuário gerada pelo openHAB que apresenta informações e permite interações, podendo ser configurada como desejar.

Fonte: OpenHAB (2021)

Para ilustrar melhor os conceitos, a Figura 6 irá apresentar como esses conceitos se comportam no openHAB.

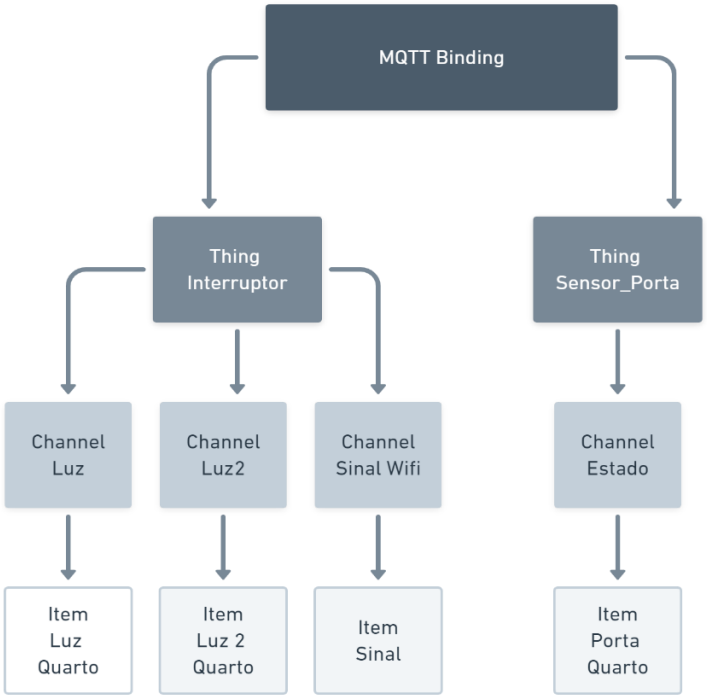

Figura 6 - Apresentação hierárquica dos conceitos openHAB
Fonte: Elaborado pelo autor

No exemplo da Figura 6, foi utilizado o MQTT Binding que é capaz de receber e enviar as mensagens pelo broker MQTT configurado, e cada Binding pode ter inúmeros Things, e cada Thing nesse caso pode representar: Uma coisa, dispositivo ou um cômodo da casa para englobar todos os dispositivos que se comunicam por meio do protocolo MQTT.

Abaixo do nível de Thing, temos o Channel que funciona como um canal mesmo, e cada Thing poderá ter também inúmeros Channels. Ao final, os Items que podem estar associado a nenhum, um ou mais canais e os Channels podem ter vários Items associados a ele.

3.2 PLANEJAMENTO DOS MÓDULOS

Antes de definir como foram feitos os módulos de controle e sensor, foi necessário realizar uma busca e análise dos principais protocolos existentes para *IoT* e micro controladores.

O grande problema ao escolher os protocolos *ZigBee* ou *Z-Wave*, é o fato de ter que desenvolver um gateway para traduzir os dados da rede local para a rede dos dispositivos e vice-versa. Após a análise, apresentou-se inviável o uso destes protocolos por elevar o custo do projeto desenvolver um *gateway* e ter que adicionar componentes nos módulos para serem compatíveis com o padrão de comunicação.

Buscando novamente formas de se desenvolver os módulos que surgiram as opções do protocolo MQTT e os micro controladores da família ESP, que os mais utilizados atualmente são o ESP8266 e o ESP32.

A escolha foi do ESP8266 por possuir um custo reduzido em relação ao seu irmão mais novo, e atender perfeitamente o que se deseja nos módulos a serem desenvolvidos. De acordo com a Espressif Systems, este módulo tem a capacidade de adicionar no mínimo sete componentes externos através de seus pinos de entrada e saída, chamados de GPIO (*General Purpose Input Output*).

Outro motivo da escolha desse módulo é a facilidade de realizar a sua programação que é muito parecida com Arduino e podendo até utilizar da mesma IDE, que conta com uma comunidade muito ativa no desenvolvimento de novas bibliotecas para adição de novos dispositivos e ferramentas. E uma dessas bibliotecas é a *pubsubclient*, que é uma biblioteca muito leve e oferece suporte aos módulos que foram construídos a comunicação por meio do protocolo MQTT, além de também ser facilmente encontrada e instalada na própria IDE do Arduino.

O principal protocolo deste projeto será o MQTT, por meio dele que todas as informações dos módulos serão transmitidas e também por onde os comandos serão enviados. Por ser totalmente compatível com o chip ESP8266, não é necessário adicionar novos componentes para ter a comunicação, pelo fato de utilizar a própria rede Wi-Fi presente no chip como meio de tráfego dos dados.

3.2.1 Módulo sensor de porta ou janela

Para a construção desse módulo, foram realizadas pesquisas de dispositivos similares em que se encaixariam no projeto e foi encontrado um projeto de eletrônica feito por Kaczynski (2017) que corresponde à um sensor de porta baseado no chip ESP8266 com as características necessárias para o projeto em desenvolvimento.

No projeto desenvolvido por Kaczynski (2017) foram disponibilizados arquivos Gerber para impressão da placa de circuitos eletrônicos (PCB), além da lista de componentes para a soldagem na placa. A impressão dessa placa foi encomendada numa empresa do ramo de impressão de PCB's.

Para este módulo, toda a programação foi feita através da IDE do Arduino, utilizando de bibliotecas fornecidas pela própria plataforma que são compatíveis com o chip ESP8266.

A lógica de funcionamento deste módulo é bastante simples: Ao detectar uma alteração no estado do sensor da porta onde está instalado, o código da biblioteca *pubsubclient* é chamado para fazer uma publicação no tópico definido como *publish:* "Quarto-Renan/Porta/state". Essa publicação irá aparecer no servidor *broker* MQTT, que por sua vez permitirá integrar ao openHAB, e irá captar e alterar o estado do item que está "escutando" esse tópico.

3.2.2 Módulo interruptor simples

Inicialmente para este módulo, era planejada a construção total do mesmo com baseando-se em projetos existentes, seguindo os moldes do anterior. Entretanto, após observar soluções já existentes no mercado, a marca chinesa ITEAD apresentou-se com ótimos dispositivos, especialmente para esse caso, o modelo Sonoff Basic.

Uma construção de um dispositivo similar iria sair com um custo mais elevado e não conseguiria ter um tamanho compacto como o modelo da ITEAD, por ser um produto comercial já testado pelo segmento de mercado especialista.

O ponto negativo desse equipamento é a obrigatoriedade de utilização do aplicativo da própria ITEAD, o que torna difícil a comunicação com a central de

controle que foi planejada, permitindo apenas o controle através da internet pelo *software* proprietário da empresa. Entretanto, a principal característica que o Sonoff Basic possui, é de ele ser baseado no chip ESP8266, motivo pelo qual o torna possível a integração dele no sistema de automação residencial proposto.

Por ele ser baseado no chip da Espressif Systems, abre-se a possibilidade de reprogramá-lo de acordo com as necessidades que buscamos, alterando o *firmware*[3] de fábrica pré-instalado. Com a função de auxiliar neste processo, o Tasmota que se trata de um projeto de *firmware* de código aberto para dispositivos ESP8266, será utilizado para modificar o Sonoff Basic.

De acordo com o Tasmota, ao atualizar o *firmware*, o dispositivo deixará de depender da internet e passará a funcionar apenas na rede local, além de receber uma capacidade de se comunicar por MQTT à central openHAB para receber comandos e enviar mensagens declarando o seu estado no momento. Somado a isso, o novo *firmware* permite agora o dispositivo que receba um sensor externo conectado à uma de suas portas GPIO, podendo ser: Sensor de temperatura, luminosidade ou até mesmo infravermelho, de acordo com as necessidades que o usuário encontrar.

3.2.3 Módulo interruptor de iluminação

O controle da iluminação do ambiente é fundamental para uma automação residencial, e o módulo designado para essa tarefa deve cumprir esse papel funcional mas deve também ter uma estética que colabore com a aparência do cômodo. Seguindo a mesma ideia do módulo de interruptor utilizado para o ventilador, a construção de um módulo que atenda as expectativas e que consiga um custo baixo não seria possível em comparação com também um modelo específico para essa funcionalidade da ITEAD, chamado Sonoff TX. Existem algumas variações do modelo, mas para o projeto, foi escolhido o modelo T0 na cor branca com duas teclas de acionamento.

3 Micro programas de baixo nível para uso específico, utilizados geralmente em dispositivos com baixa capacidade de processamento para controle direto ao hardware. ((BERG, 1980))

Figura 7 - Modelos de interruptores Sonoff TX
Fonte: (ITEAD INTELLIGENT SYSTEMS, 2021)

Assim como o modelo Sonoff Basic, o Sonoff TX é baseado no ESP8266 e possibilita a sua reprogramação de *firmware*, sendo totalmente compatível com a instalação do Tasmota. Da mesma forma que o módulo anterior, é feita a alteração do *firmware,* habilitando o chip a se comunicar por MQTT com o servidor *broker* instalado na central openHAB, o tornando totalmente integrado ao sistema de automação.

3.2.4 Módulo LED RGB com sensor de temperatura

Após pesquisas de como é possível realizar o controle de uma fita de LED RGB através do chip ESP8266, novamente o projeto de Kaczynski (2017) apresentou-se com uma solução bastante interessante de um módulo com as características necessárias para a função desejada. O módulo em questão é um controlador de fitas de LED RGB baseado no chip da Espressif Systems, assim como no módulo sensor de porta e janela, foi disponibilizado um arquivo para impressão da placa de circuitos além da lista de componentes necessários para a soldagem e um passo a passo de instalação inicial.

Por padrão esse módulo permite apenas o controle dos LEDs, porém para o projeto deste trabalho seria interessante adicionar um sensor de temperatura e umidade, apresentado anteriormente, o sensor escolhido foi o DHT11 da empresa Aosong Eletronics.

O *firmware* utilizado é similar ao módulo de sensor de portas, adicionando funcionalidades para controle da cor do LED, em que é recebida a cor no padrão de cores RGB por MQTT no tópico definido para *subscribe*, por sua vez é

convertido no código para valor de 0 a 1023 para cada canal individualmente, onde 0 é desligado e 1023 é o máximo de brilho do LED.

Para realizar a leitura do sensor de temperatura, a biblioteca de código aberto DHT-sensor-library desenvolvida pela Adafruit é facilmente encontrada na IDE do Arduino e foi utilizada para a integração deste sensor no código.

3.3 TORNANDO MAIS INTELIGENTE

Poder controlar todos os dispositivos na palma da mão é um grande passo na automação residencial, mas poder automatizar algumas tarefas faz toda a diferença, e é essa a característica que deixa a casa mais inteligente.

A central de controle deste projeto, com o openHAB é capaz de realizar essas tarefas de forma automática com o conceito de Rules, ou regras, anteriormente introduzido.

Essas regras podem ser arquivos de extensão ".rules" criados na pasta de configuração do openHAB ou podem ser criados pela interface gráfica de uma forma mais amigável para quem não possui experiência com programação. E a própria OpenHAB (2021) define como "Regras são usadas para automatizar processos: cada regra pode ser acionada, o que invoca um script que executa qualquer tipo de tarefa, por exemplo, acenda as luzes, modifique seus Itens, faça cálculos matemáticos, inicie cronômetros etc.".

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

Figura 8 - Sintaxe de uma regra escrita
Fonte: OpenHAB, 2021

Como pôde ser vista na figura anterior, cada regra deve ter um nome único e sua estrutura funciona como uma condição, caso alguma coisa aconteça então execute o bloco abaixo. Essas condições podem ser: Estado de um item

alterado, item recebeu um comando, agendamento de tarefas por CronJob[4] e etc.

Alguns exemplos de aplicação das regras: Todos os dias às 17 horas ligue as luzes do jardim, quando o sol nascer apague as luzes, se o portão for aberto à noite acenda a luz da garagem, se a porta for aberta me envie uma notificação no celular. São praticamente infinitas as possibilidades de regras para automatizar, fica a cargo da criatividade e da disponibilidade de dispositivos para executar os comandos.

4 Agendador de tarefas automáticas baseado em tempo, encontrado em sistemas operacionais semelhantes ao Unix. (GARBARINO, 2019)

4 RESULTADOS OBTIDOS

O sistema desse projeto é um modelo bem simples e fácil de ser compreendido, com base na arquitetura já presente no mundo atual de *IoT* e automação residencial. A Figura 9 apresenta a arquitetura do sistema obtido como resultado deste trabalho:

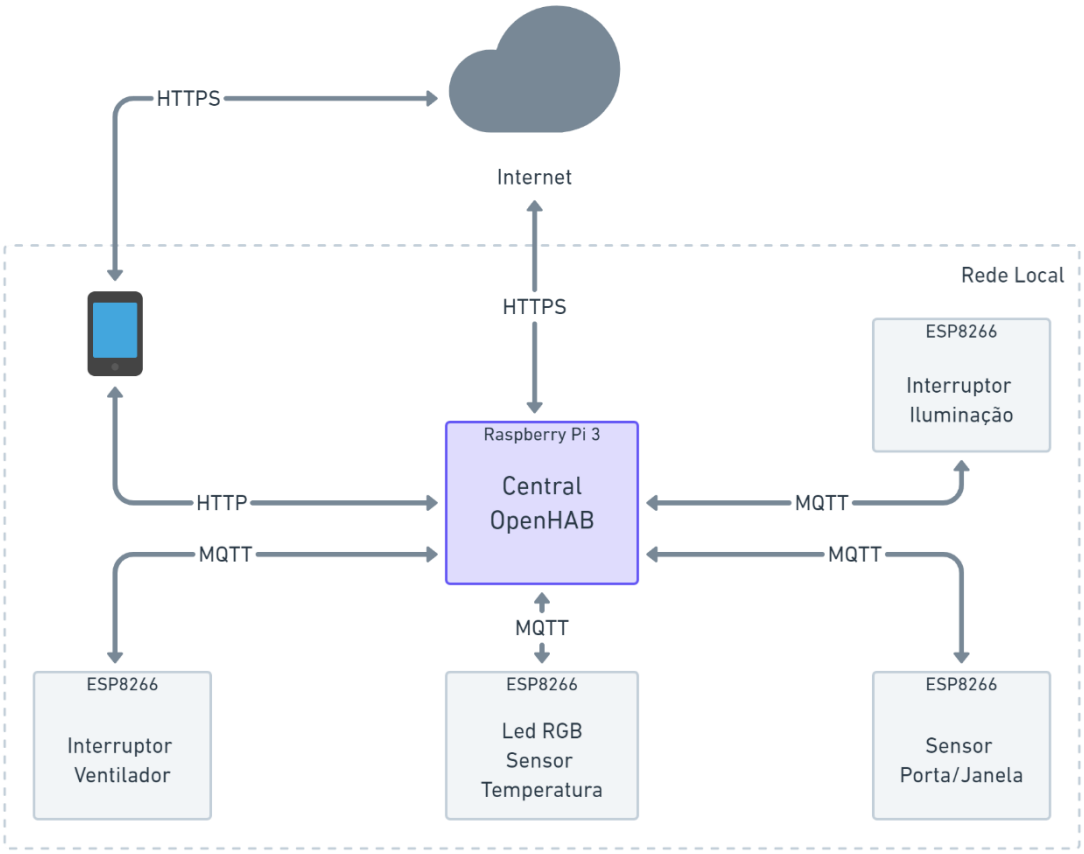

Figura 9 - Arquitetura do sistema desenvolvido
Fonte: Elaborado pelo autor

4.1 MÓDULOS

4.1.1 Módulo sensor de porta

Depois de concluída a soldagem dos componentes, o resultado do módulo de sensor de porta pode ser visto na Figura 10:

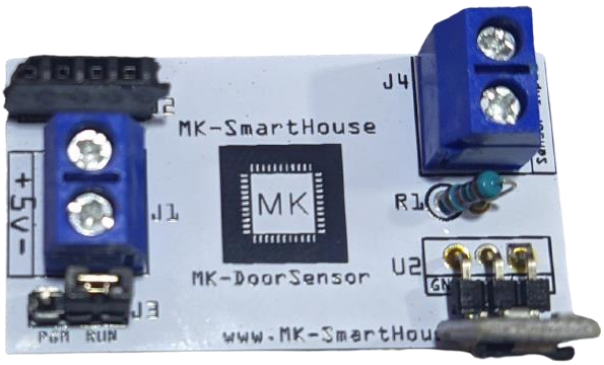

Figura 10 - Resultado do módulo sensor para portas ou janelas
Fonte: Elaborado pelo autor

Neste módulo, em um dos terminais de cor azul onde é conectado o sensor magnético de contato, na outra extremidade, é a entrada de energia 5 *volts* que pode ser utilizada qualquer carregador de celular ou portas USB de computador.

Após a programação pela IDE do Arduino, o módulo foi capaz de se conectar à rede local sem problemas e também ao *broker* MQTT para o envio do estado a atual em que se encontra o sensor.

4.1.2 Módulo interruptor simples para ventilador

Concluído os passos do guia instalação do Tasmota, e feito as configurações da rede *wireless* e MQTT, o módulo foi instalado para controlar um ventilador de mesa. Na figura a seguir, o Tasmota oferece uma página de controle do dispositivo que é acessada pelo IP atribuído a ele na rede.

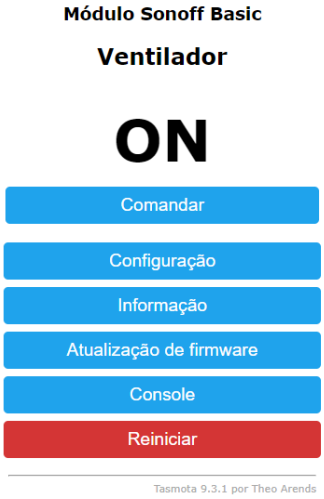

Figura 11 - Página de configuração do dispositivo
Fonte: Elaborado pelo autor

A página de configuração permite controlar totalmente o dispositivo, como por exemplo: Alterar a rede *wireless* que deseja se conectar, configurar endereço do

broker MQTT, definir tópicos de *publish* e *subscribe* para comandar o dispositivo e também realizar a atualização de firmware via rede para uma versão mais nova do Tasmota.

4.1.3 Módulo interruptor de iluminação

Finalizada a instalação e configuração do interruptor, a central openHAB estará capaz de comandar as lâmpadas e receber o estado atual caso alguma tecla seja acionada fisicamente.

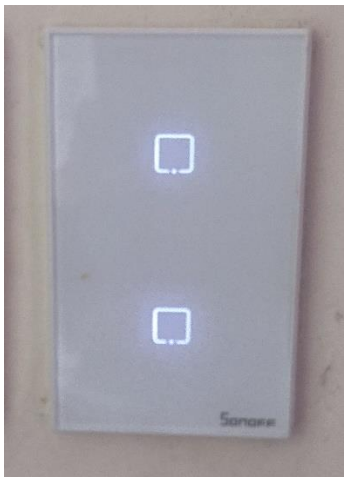

Figura 12 - Interruptor Sonoff TX de duas teclas instalado
Fonte: Elaborado pelo autor

Assim como em todos os dispositivos em que é instalado o *firmware* Tasmota, a página de configuração também está presente para realizar ações administrativas individualmente no dispositivo.

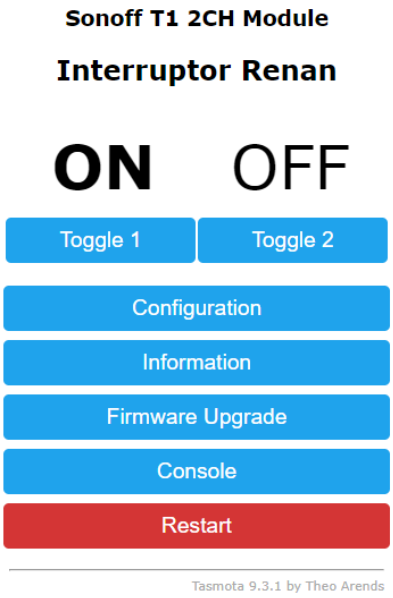

Figura 13 - Página de configuração do módulo interruptor de iluminação

Fonte: Elaborado pelo autor

4.1.4 Módulo LED RGB com sensor de temperatura

Após a soldagem dos componentes à placa de circuitos impressa, a próxima figura apresenta os resultados do módulo:

Figura 14 - Módulo de controle da fita RGB concluído
Fonte: Elaborado pelo autor

Terminada a programação e integração do sensor, o módulo está apto a receber comandos para controle dos LEDs, e enviar o valor medido pelo sensor de temperatura e umidade do ambiente.

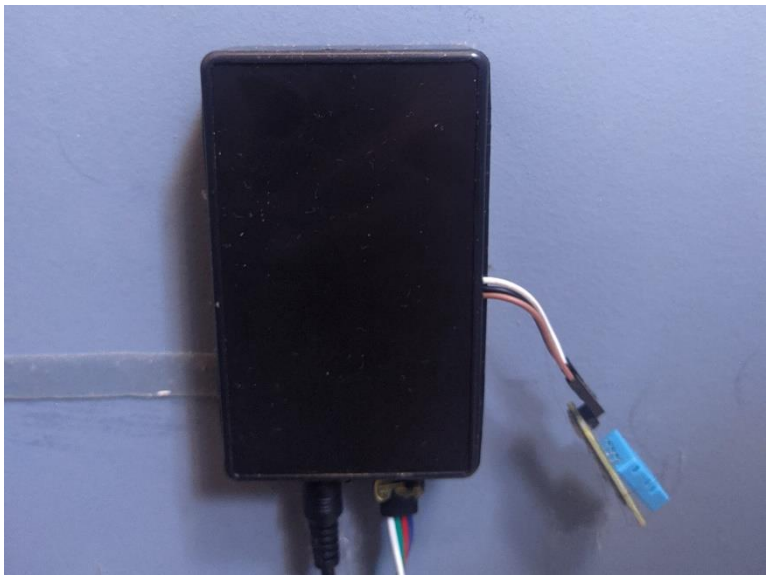

Figura 15 - Módulo RGB instalado com sensor de temperatura adaptado
Fonte: Elaborado pelo autor

4.2 REGRAS CRIADAS PARA A AUTOMAÇÃO DAS TAREFAS

Neste projeto, as regras foram criadas de forma textual através dos arquivos de configuração, tendo assim um controle maior sobre a regra que está sendo escrita.

A primeira regra será para controle de cenas, por exemplo, um modo filme em que ao ativado a luz principal se apaga e apenas a luz secundária e a fita de LED são acesas:

```
rule "Modo filme"
when
  Item Quarto_Cena changed to 1
then
  Quarto_Luz.sendCommand(OFF)
  Thread::sleep(500)
  Quarto_Luz2.sendCommand(ON)
  FitaLedColor.sendCommand(ON)
end
```

Figura 16 - Regra para criação de cena modo filme
Fonte: Elaborado pelo autor

A próxima regra realiza uma integração com reprodutor de mídia chamado Kodi, que se trata de uma central de entretenimento de código aberto com muitas funcionalidades, e o openHAB possui um Binding capaz de obter todas as informações em tempo real do que está sendo reproduzido e também enviar comandos de pause, play, aumentar volume e entre outros. Esta regra irá também apagar ou acender as luzes caso o vídeo seja reproduzido ou pausado, apenas se a cena selecionada é a de filme.

```
rule "Apaga as luzes ao dar play"
when
    Item KodiRenan_control changed
then
  if(Quarto_Cena.state == 1){
    if(KodiRenan_control.state == PAUSE)//Acende ao pausar
    {
      Quarto_Luz.sendCommand(ON)
      Thread::sleep(500)//Aguarda meio segundo
      Quarto_Luz2.sendCommand(OFF)
      FitaLedColor.sendCommand(OFF)
    }
    if(KodiRenan_control.state == PLAY)//Apaga ao acender
    {
      Quarto_Luz.sendCommand(OFF)
      Thread::sleep(500)//Aguarda meio segundo
      Quarto_Luz2.sendCommand(ON)
      FitaLedColor.sendCommand(ON)
    }
  }
end
```

Figura 17 - Automatização das luzes ao reproduzir um vídeo
Fonte: Elaborado pelo autor

A terceira automatização será para controlar as luzes, e a mídia em reprodução caso a porta do quarto seja aberta ou fechada. No caso da porta se abrir, as luzes irão acender e a reprodução será pausada. Se a porta for fechada, as luzes se apagam e a mídia em questão volta a ser reproduzida. As duas situações só irão ocorrer caso a cena filme esteja selecionada.

```
rule "Pausa a reproducao se abrir a porta"
when
    Item Quarto_Porta changed
then
  if(Quarto_Porta.state == OPEN && Quarto_Cena.state == 1){
      Quarto_Luz.sendCommand(ON)
      Quarto_Luz2.sendCommand(OFF)
      KodiRenan_control.sendCommand(PAUSE)
  }
  if(Quarto_Porta.state == CLOSED && Quarto_Cena.state == 1){
      Quarto_Luz.sendCommand(OFF)
      Quarto_Luz2.sendCommand(ON)
      KodiRenan_control.sendCommand(PLAY)
  }
end
```

Figura 18 - Automatização de tarefas caso a porta mude de estado
Fonte: Elaborado pelo autor

Através dessas regras criadas foi possível obter uma automatização bem interessante e deixar um pouco mais inteligente a automação do projeto. Com

inúmeras possibilidades, as regras podem praticamente fazer quase tudo com os dispositivos por meio de uma programação simples baseada em condicionais e eventos que são disparados para executa-las.

4.3 CENTRAL DE CONTROLE OPENHAB

Os resultados da central de controle que está executando a plataforma openHAB foram excelentes, apresentando um grande nível de confiabilidade no sistema mesmo em situações onde é desligado da energia elétrica, o mesmo retorna para o estado anterior de todos os dispositivos exibindo os mesmos valores de momentos antes ao desligamento.

Durante o período de análise e testes do sistema no intervalo de mais de 6 meses ininterruptos, a central que é executada na Raspberry PI 24 horas por dia, permaneceu estável sem a necessidade de reinicializações no sistema operacional.

A integração com o protocolo MQTT é bastante eficiente quanto ao tempo de resposta dos comandos para os módulos, praticamente de forma instantânea é o acionamento pela rede local.

As interfaces gráficas do sistema para controle via navegador ou aplicativo móvel se mostraram simples de configurar seguindo o guia inicial da plataforma:

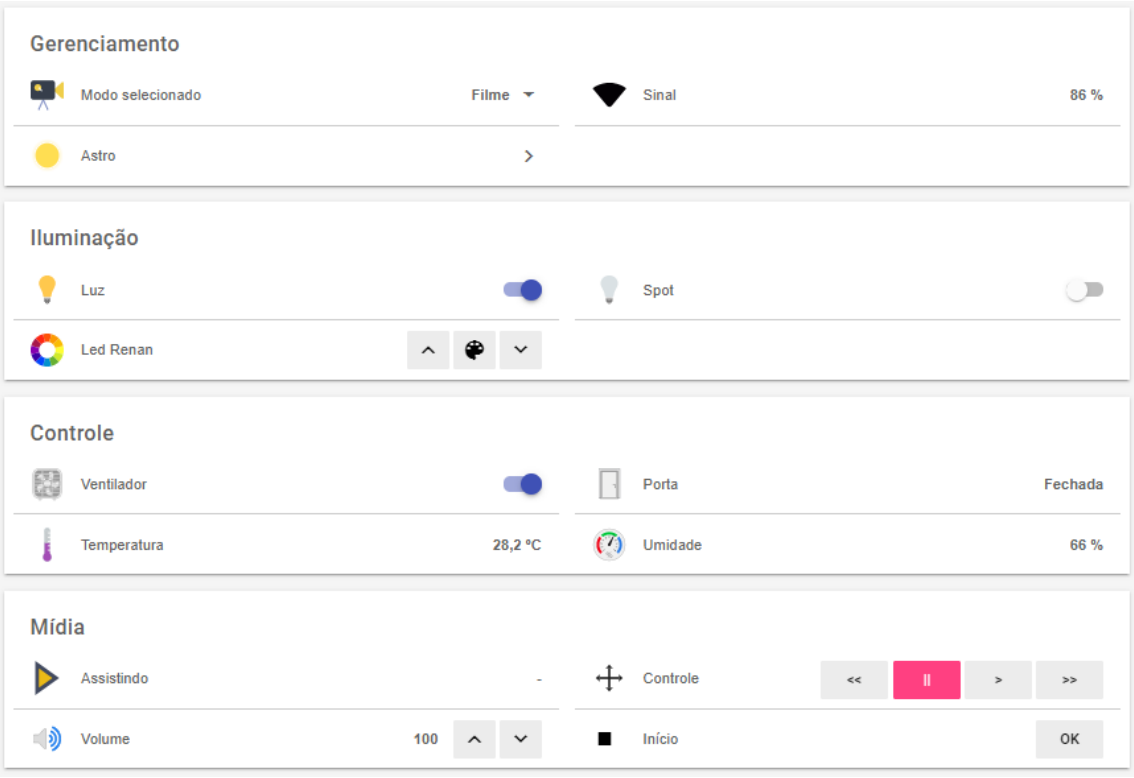

Figura 19 - Interface do openHAB acessada por navegador
Fonte: Elaborado pelo autor

Por meio das interfaces é possível controlar todos os dispositivos, ver o estado atual, realizar o acionamento de cenas e também controlar a mídia que está sendo executada no aplicativo central de multimídia.

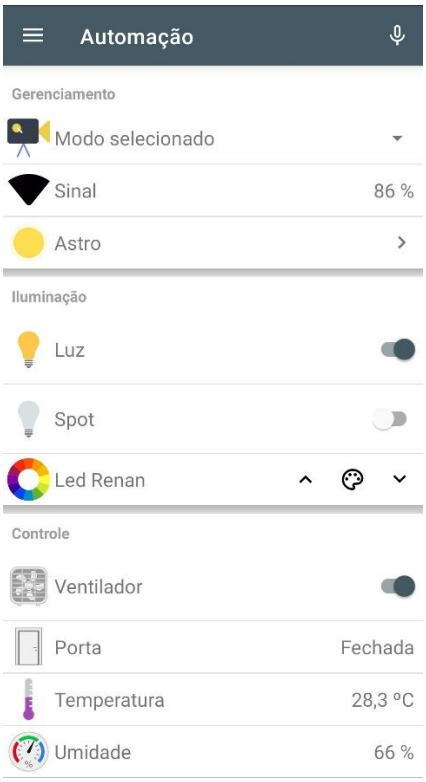

Figura 20 - Interface mobile do aplicativo openHAB
Fonte: Elaborado pelo autor

A interface mobile se apresentou muito similar à interface web no funcionamento e acionamento dos dispositivos com uma boa usabilidade e interatividade.

4.4 CUSTO FINAL DO PROJETO

A tabela a seguir mostra o custo final do projeto, especificando o valor de cada módulo individualmente:

Tabela 2 - Custo total do projeto

Equipamento	Valor
Kit completo Raspberry PI 3B	R$ 351,95
Módulo sensor de porta ou janela	R$ 24,30
Módulo interruptor simples para ventilador	R$ 41,80
Módulo interruptor para iluminação	R$ 92,45
Módulo controle RGB com fita LED e sensor de temperatura	R$ 89,65
Total do projeto	R$ 600,15

Fonte: Elaborado pelo autor

5 CONCLUSÕES

Inicialmente, a ideia deste projeto foi ser aplicado em uma área menor, que no caso é um quarto, a fim de experimentar o funcionamento desses módulos e da central para posteriormente ser expandido para outros cômodos e assim ser implantado em uma casa inteira. Através da arquitetura deste projeto, para uma residência precisa-se de apenas uma central e em sequência serem adicionados novos módulos de controle e sensores à medida que forem necessários, tornando uma experiência de automação residencial ainda mais completa.

Os resultados obtidos foram animadores, permitindo o controle total sobre os módulos com um tempo de resposta baixo sem a necessidade do uso de aplicações na nuvem de terceiros. Soluções de código aberto produzidas pelas comunidades na internet tornaram mais simples a integração dos módulos, e flexibilizaram o uso dos dispositivos produzidos por empresas que de certa forma limitam o uso dos mesmos.

Comparado aos sistemas de automação de empresas consolidadas no mercado, o custo obtido neste projeto esteve inferior, viabilizando uma automação residencial simples com um baixo investimento inicial, e de acordo com a demanda e orçamento, poderá evoluir com a adição de novos módulos.

Entretanto, o desenvolvimento deste projeto é necessário um conhecimento básico nas áreas de programação e eletrônica, além de praticamente todo o conteúdo disponível ser na língua inglesa, ocasionando uma dificuldade de democratizar o acesso ao mundo de automação residencial. A fim de contornar essa situação, as recentes atualizações das plataformas de automação estão cada vez mais voltadas para o usuário “comum”.

Seguindo essa tendência, é provável que em poucos anos, a integração por meio de um protocolo em comum tornará ainda mais simples, acessível e com uma variedade muito maior de dispositivos para a automação residencial.

REFERÊNCIAS BIBLIOGRÁFICAS

AOSONG ELETRONICS. **Guagzhou Aosong Eletronics Co**. Disponivel em: <http://www.aosong.com/en/.>. Acesso em: 10 Jun 2020.

ARDUINO. What is Arduino? **Arduino.cc**. Disponivel em: <http://arduino.cc>. Acesso em: 20 Abril 2020.

BERG, H. K. **Correctness of Firmware -An Overview-**. Firmware Engineering. Berlim: Springer. 1980. p. 173-224.

COOK, D. J. How Smart Is Your Home?. **Science**, v. 335, Mar 2012. ISSN 0036-8075.

EMBEDDED LABORATORY. Getting Started with MQTT Using Python. **Embedded Laboratory**, 2018. Disponivel em: <http://embeddedlaboratory.blogspot.com/2018/01/getting-started-with-mqtt-using-python.html>. Acesso em: 20 Abr 2020.

ESPRESSIF SYSTEMS. ESP8266 Wroom Series. **Espressif**. Disponivel em: <https://www.espressif.com/en/products/hardware/esp-wroom-02/overview>. Acesso em: 8 Maio 2020.

FABI, V.; SPIGLIANTINI, G.; CORGNATI, S. P. Insights on Smart Home concept and occupants' interaction with building controls. **Energy Procedia**, Turim, Itália, 2016.

FRENZEL, L. What's The Difference Between ZigBee And Z-Wave? **Eletronic Design**, 2012. Disponivel em: <https://www.electronicdesign.com/communications/what-s-difference-between-zigbee-and-z-wave>. Acesso em: 15 Maio 2020.

G1.GLOBO.COM. Empresa fatura R$ 1 milhão com serviços de automação residencial. **Globo.com**, 2013. Disponivel em: <http://g1.globo.com/economia/pme/noticia/2013/05/empresa-fatura-r-1-milhao-com-servicos-de-automacao-residencial.html>. Acesso em: 18 Jun 2020.

GARBARINO, E. CronJobs. In: _____ **Beginning Kubernetes on the Google Cloud Platform**. Apress, Berkeley, CA: [s.n.], 2019.

HARBOR RESEARCH. Product developers and platform players are jockeying for a piece of the Smart home market. **Harbor Research**, 2017. Disponivel em: <http://harborresearch.com/smart-home-moving-luddites-forward>. Acesso em: 10 Maio 2020.

ITEAD INTELLIGENT SYSTEMS. ITEAD Smart Home. **Itead**, 2021. Disponivel em: <https://www.itead.cc/smart-home.html>. Acesso em: 11 Janeiro 2021.

KACZYNSKI, M. Door Sensor. **MK-SmartHouse**, 2017. Disponivel em: <https://www.mksmarthouse.com/door-sensor>. Acesso em: 17 Jun 2020.

KACZYNSKI, M. LED Strip Control. **MK SmartHouse**, 2017. Disponivel em: <https://www.mksmarthouse.com/led-control>. Acesso em: 20 Jun 2020.

KAVEN, O. Zensys' Z-Wave Technology. **PC Mag**, 2005. Disponivel em: <https://www.pcmag.com/article2/0,2817,1749559,00.asp>. Acesso em: 15 Maio 2020.

LACERDA, F.; LIMA-MARQUES, M. Da necessidade de princípios de Arquitetura da Informação para a Internet das Coisas. **ResearchGate**, 2015. Disponivel em: <https://www.researchgate.net/publication/282352769_Da_necessidade_de_principios_de_Arquitetura_da_Informacao_para_a_Internet_das_Coisas>. Acesso em: 16 Mar 2021.

LUETH, K. L. State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating. **Iot Analytics**, 8 agosto 2018. Disponivel em: <https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/>. Acesso em: 05 Jun 2020.

MILLER, M. **My Smart Home for Seniors**. [S.l.]: Que Publishing, 2017.

O TEMPO. Projeto para ter uma casa conectada custa a partir de R$ 6.000. **O Tempo**, 2017. Disponivel em: <https://www.otempo.com.br/interessa/tecnologia-e-games/projeto-para-ter-uma-casa-conectada-custa-a-partir-de-r-6-000-1.1473984>. Acesso em: 18 Jun 2020.

OPENHAB. **openHAB.org**. Disponivel em: <https://openhab.org>. Acesso em: 10 Mar 2021.

PASTORE, P. O que é um protocolo? **UFRJ**, 2003. Disponivel em: <https://www.gta.ufrj.br/grad/03_1/http/protoc.html>. Acesso em: 10 Abr 2021.

RASPBERRY PI. **Raspberry Pi Foundation**. Disponivel em: <https://www.raspberrypi.org>. Acesso em: 26 Maio 2020.

ROBLES, R. J.; KIM, T.-H. Applications, Systems and Methods in Smart Home Technology: A Review. **International Journal of Advanced Science and Technology**, Daejon, Korea, Fev 2010.

SONGLE. **Ningbo Songle Relay Co., Ltd**. Disponivel em: <http://www.songle.com>. Acesso em: 10 Jun 2020.

SURE UNIVERSAL. SURE Plataform - Complete IoT Ecosystem. **Sure Universal**, 2019. Disponivel em: <https://www.sureuniversal.com>. Acesso em: 10 Jun 2020.

TASMOTA. Alternative firmware for ESP8266 and ESP32 based devices. **GitHub**, 2021. Disponivel em: <https://github.com/arendst/tasmota/>. Acesso em: 18 Mai 2020.

WAHLBERG, A. Smart-Home Technology for the Disability Community. **Connector Supplier**, 2018. Disponivel em: <https://www.connectorsupplier.com/smart-home-technology-for-the-disability-community/>. Acesso em: 05 Jun 2020.

YUAN, M. Conhecendo o MQTT. **IBM**, 2017. Disponivel em: <https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html>. Acesso em: 20 Abr 2020.

APÊNDICES

APÊNDICE A – Código fonte do módulo sensor de porta

```
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <ESP8266WiFi.h> // Importa a Biblioteca ESP8266WiFi
#include <PubSubClient.h> // Importa a Biblioteca PubSubClient

//defines:
//defines de id mqtt e tópicos para publicação e subscribe
#define TOPICO_SUBSCRIBE "Quarto-Renan/Porta"     //tópico MQTT de escuta
#define TOPICO_PUBLISH "Quarto-Renan/Porta/state"    //tópico MQTT de envio de
informações para Broker
#define ID_MQTT  "SensorPortaRenan"

// WIFI
const char* SSID = ""; // SSID / nome da rede WI-FI que deseja se conectar
const char* PASSWORD = ""; // Senha da rede WI-FI que deseja se conectar

// MQTT
const char* BROKER_MQTT = "192.168.1.50"; //URL do broker MQTT que se deseja utilizar
int BROKER_PORT = 1883; // Porta do Broker MQTT

//////////////Sensor Porta/Janela////////////////
long unsigned int lowIn;

//tempo em millisegundos que sensor tem de ficar low
long unsigned int pause = 100;

//sensor variaveis
boolean lockLow = true;
boolean takeLowTime;

int sensorPin = 13;

//Variáveis e objetos globais
WiFiClient espClient; // Cria o objeto espClient
PubSubClient MQTT(espClient); // Instancia o Cliente MQTT passando o objeto espClient

//Prototypes
void initWiFi();
void initMQTT();
void reconectWiFi();
//void mqtt_callback(char* topic, byte* payload, unsigned int length);
void VerificaConexoesWiFIEMQTT(void);

 /*  Implementações das funções
 */
void setup()
{
   ArduinoOTA.setPassword((const char *)"senha");
```

```
    //inicializações:
    initWiFi();
    initMQTT();

    pinMode(sensorPin, INPUT);//Pino Sensor
    digitalWrite(sensorPin, LOW);

    //Upload via Wifi OTA//////////////////////
    WiFi.mode(WIFI_STA);
    WiFi.hostname("SensorPorta_Renan");
    WiFi.begin(SSID, PASSWORD);

    while (WiFi.waitForConnectResult() != WL_CONNECTED) {
      delay(5000);
      ESP.restart();
    }
    ArduinoOTA.begin();

}

void initWiFi()
{
    reconectWiFi();
}

void initMQTT()
{
    MQTT.setServer(BROKER_MQTT, BROKER_PORT);   //informa qual broker e porta deve
ser conectado
//    MQTT.setCallback(mqtt_callback);            //atribui função de callback (função chamada
quando qualquer informação de um dos tópicos subescritos chega)
}

void reconnectMQTT()
{
    while (!MQTT.connected())
    {
        if (MQTT.connect(ID_MQTT))
        {
            MQTT.subscribe(TOPICO_SUBSCRIBE);
        }
        else
        {
            delay(2000); //falha ao conectar, tenta novamente em 2s
        }
    }
}

//Função: reconecta-se ao WiFi
void reconectWiFi()
{
    //se já está conectado a rede WI-FI, nada é feito.
```

```
    //Caso contrário, são efetuadas tentativas de conexão
    if (WiFi.status() == WL_CONNECTED)
        return;

    WiFi.begin(SSID, PASSWORD); // Conecta na rede WI-FI

    while (WiFi.status() != WL_CONNECTED)
    {
        delay(100);
    }

}

void VerificaConexoesWiFIEMQTT(void)
{
    if (!MQTT.connected())
        reconnectMQTT(); //se não há conexão com o Broker, a conexão é refeita

     reconectWiFi(); //se não há conexão com o WiFI, a conexão é refeita
}

//programa principal
void loop()
{
    //garante funcionamento das conexões WiFi e ao broker MQTT
    VerificaConexoesWiFIEMQTT();
    //Mantem comunicação com Broker
    //MQTT.loop();

    if(digitalRead(sensorPin) == HIGH)
    {
      if(lockLow)
      {
        //verifica e aguarda para mudar o estado para LOW antes de fazer qualquer saída:
        lockLow = false;
        MQTT.publish(TOPICO_PUBLISH, "OPEN");
        delay(50);
      }
      takeLowTime = true;
    }

    if(digitalRead(sensorPin) == LOW)
    {
      if(takeLowTime)
      {
        lowIn = millis();          //guarda o tempo de transição de HIGH para LOW
        takeLowTime = false;       //certifica de que so seja feito apenas no início do LOW
      }
      //se o sensor está LOW por mais tempo do que a pausa dada,
      //presumimos que nenhuma detecção mais acontecerá
      if(!lockLow && millis() - lowIn > pause)
      {
        //garante que este bloco de código só seja executado novamente após
```

```
        //uma nova detecção de movimento
        lockLow = true;
        MQTT.publish(TOPICO_PUBLISH, "CLOSED");
        delay(50);
      }
    }
}
```

APÊNDICE B – Código fonte do módulo controlador fita LED RGB

```
#include <ESP8266WiFi.h> // Importa a Biblioteca ESP8266WiFi
#include <PubSubClient.h> // Importa a Biblioteca PubSubClient

#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <ESP8266HTTPUpdateServer.h>

#include <DHT.h> // Importa Biblioteca do sensor de temperatura e umidade DHT11

//defines de id mqtt e tópicos para publicação e subscribe
#define TOPICO_SUBSCRIBE "Quarto-Renan/Led"     //tópico MQTT de escuta
#define ID_MQTT  "Led-Renan"

//Define DHT 11
#define DHTPIN 10
#define DHTTYPE DHT11 //Tipo do sensor DHT11

//MQTT
String mqtt_topic = "Quarto-Renan/Led/state";    //tópico MQTT de envio de informações para
Broker

// WIFI
const char* SSID = "rede"; // SSID / nome da rede WI-FI que deseja se conectar
const char* PASSWORD = "senha"; // Senha da rede WI-FI que deseja se conectar

//HTTP UPDATE
// Host Name of Device
const char* host = "Led-Renan";
//
const char* update_path = "/firmware";
// Username to access the web update page
const char* update_username = "admin";
// Password to access the web update page
const char* update_password = "admin";

// MQTT
const char* BROKER_MQTT = "192.168.1.50"; //URL do broker MQTT que se deseja utilizar
int BROKER_PORT = 1883; // Porta do Broker MQTT

//Portas de cada canal RGB
const int greenLed = 12;
const int redLed = 14;
const int blueLed = 13;

int r = 0;
int g = 0;
int b = 0;
int r2 = 0;
int g2 = 0;
int b2 = 0;
```

```
// inicializa valor dos leds
String greenLedValS = "0";
String redLedValS = "0";
String blueLedValS = "0";

// inicializa valor dos leds
int greenLedVal = 0;
int redLedVal = 0;
int blueLedVal = 0;

// inicializa valor dos leds
int greenLedValTemp = 0;
int redLedValTemp = 0;
int blueLedValTemp = 0;

// inicializa valor dos leds
String greenLedValLast = "0";
String redLedValLast = "0";
String blueLedValLast = "0";

int lowestNumber;
int aSpeed = 100;
//effect Variable
enum Effects { SETCOLOR };
Effects currentEffect = NONE;
//Variáveis e objetos globais

ESP8266WebServer httpServer(80);//Cria servidor http
ESP8266HTTPUpdateServer httpUpdater;//Cria servidor para update

WiFiClient espClient; // Cria o objeto espClient
PubSubClient MQTT(espClient); // Instancia o Cliente MQTT

DHT dht(DHTPIN, DHTTYPE); //Inicializando o objeto dht do tipo DHT passando como
parâmetro o pino (DHTPIN) e o tipo do sensor (DHTTYPE)

//Prototypes
void initSerial();
void initWiFi();
void initMQTT();
void reconectWiFi();
void mqtt_callback(char* topic, byte* payload, unsigned int length);
void VerificaConexoesWiFIEMQTT(void);
void InitOutput(void);

unsigned long lastMillis = 0;

float temperatura; //variável para armazenar a temperatura
float umidade; //Variável para armazenar a umidade
//Intervalo de tempo entre leituras
const long intervalo = 60000;
//Armazena o valor (tempo) da ultima leitura
unsigned long previousMillis = 0;
```

```
void setup()
{
  pinMode(16, INPUT);
  //inicializações:
  initSerial();
  initWiFi();
  initMQTT();

  dht.begin(); //Inicializa o sensor DHT11

  //Configurando WIFI
  WiFi.mode(WIFI_STA);
  WiFi.hostname("Led-Renan");
  WiFi.begin(SSID, PASSWORD);

  //Verificar conexão
  connect();

  //Configurar servidor HTTP updater
  MDNS.begin(host);
  httpUpdater.setup(&httpServer, update_path, update_username, update_password);
  httpServer.begin();
  MDNS.addService("http", "tcp", 80);
}

void connect(){
  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
    delay(5000);
    ESP.restart();
  }
}

void initSerial()
{
    Serial.begin(115200);
}

void initWiFi()
{
    reconectWiFi();
}

void initMQTT()
{
    MQTT.setServer(BROKER_MQTT, BROKER_PORT);   //informa qual broker e porta deve
ser conectado
    MQTT.setCallback(mqtt_callback);            //atribui função de callback (função chamada
quando qualquer informação de um dos tópicos subescritos chega)
}

void mqtt_callback(char* topic, byte* payload, unsigned int length)
{
```

```
  String msg;
  //obtem a string do payload recebido
  for(int i = 0; i < length; i++)
  {
     char c = (char)payload[i];
     msg += c;
  }

  if(strcmp(topic,TOPICO_SUBSCRIBE) == 0)
  {
    String stringOne = msg;

    // find first ; in the string
    int firstClosingBracket = stringOne.indexOf(';')+1;

    // find second ; in the string
    int secondOpeningBracket = firstClosingBracket + 1;
    int secondClosingBracket = stringOne.indexOf(';', secondOpeningBracket);

    // find the third ; in the string
    int thirdOpeningBracket = secondClosingBracket + 1;
    int thirdClosingBracket = stringOne.indexOf(';', thirdOpeningBracket);

    // using the locations of ; find values
    greenLedValS = stringOne.substring(0 , (firstClosingBracket - 1));
    redLedValS = stringOne.substring(firstClosingBracket , secondClosingBracket);
    blueLedValS = stringOne.substring((secondClosingBracket +1) , thirdClosingBracket);

    if ((blueLedValS != blueLedValLast) || (greenLedValS != greenLedValLast) || (redLedValS !=
redLedValLast))
    {
      r2 = (redLedValS.toInt());
      g2 = (greenLedValS.toInt());
      b2 = (blueLedValS.toInt());

      blueLedVal = blueLedValS.toInt() * 10.23;
      redLedVal = redLedValS.toInt() * 10.23;
      greenLedVal = greenLedValS.toInt() * 10.23;

      greenLedValTemp = greenLedVal;
      redLedValTemp = redLedVal;
      blueLedValTemp = blueLedVal;

      currentEffect = SETCOLOR;

      greenLedValLast = greenLedVal;
      redLedValLast = redLedVal;
      blueLedValLast = blueLedVal;
    }

    MQTT.publish("Quarto-Renan/Led/state/color", "recebida cor");
  }
}
```

```
int divisor()
{
  return min(min(blueLedVal, redLedVal), greenLedVal);
}

boolean isValidNumber(String str)
{
   boolean isNum=false;
   if(!(str.charAt(0) == '+' || str.charAt(0) == '-' || isDigit(str.charAt(0)))) return false;

   for(byte i=1;i<str.length();i++)
   {
     if(!(isDigit(str.charAt(i)) || str.charAt(i) == '.')) return false;
   }
   return true;
}

void setColor(int red, int green, int blue) {
  while ( r != red || g != green || b != blue ) {
    if ( r < red ) r += 1;
    if ( r > red ) r -= 1;

    if ( g < green ) g += 1;
    if ( g > green ) g -= 1;

    if ( b < blue ) b += 1;
    if ( b > blue ) b -= 1;

    _setColor();
    delay(10);
  }
}

void _setColor() {
  analogWrite(redLed, (r * 10.23));
  analogWrite(greenLed, (g * 10.23));
  analogWrite(blueLed, (b * 10.23));
}

void reconnectMQTT()
{
  while (!MQTT.connected())
  {
    if (MQTT.connect(ID_MQTT))
    {
        MQTT.subscribe(TOPICO_SUBSCRIBE);
    }
    else
    {
        delay(2000); //falha ao conectar, tenta novamente em 2s
    }
  }
```

```
}

//Função: reconecta-se ao WiFi
//Parâmetros: nenhum
//Retorno: nenhum
void reconectWiFi()
{
  //se já está conectado a rede WI-FI, nada é feito.
  //Caso contrário, são efetuadas tentativas de conexão
  if (WiFi.status() == WL_CONNECTED)
    return;

  WiFi.begin(SSID, PASSWORD); // Conecta na rede WI-FI

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(100);
  }

}

void VerificaConexoesWiFIEMQTT(void)
{
  if (!MQTT.connected())
    reconnectMQTT(); //se não há conexão com o Broker, a conexão é refeita

  reconectWiFi(); //se não há conexão com o WiFI, a conexão é refeita
}

//programa principal
void loop()
{
  //garante funcionamento das conexões WiFi e ao broker MQTT
  VerificaConexoesWiFIEMQTT();

  unsigned long currentMillis = millis();
  //Verifica se o intervalo já foi atingido
  if (currentMillis - previousMillis >= intervalo)
  {
    //Armazena o valor da ultima leitura
    previousMillis = currentMillis;

    temperatura = dht.readTemperature();  //Realiza a leitura da temperatura
    umidade = dht.readHumidity(); //Realiza a leitura da umidade

    MQTT.publish("Quarto-Renan/Sensor/Temperatura", String(temperatura).c_str());
    MQTT.publish("Quarto-Renan/Sensor/Umidade", String(umidade).c_str());
  }

  MQTT.loop();

  httpServer.handleClient();
}
```

APÊNDICE C – Arquivo de definição de Things: quarto.things

```
Bridge mqtt:broker:Broker [ host="192.168.1.50", secure=false, port="1883"] {
  Thing topic QuartoRenan "Quarto Renan" @ "Casa" {
    Channels:
        Type switch : Luz  "Luz" [ stateTopic="Quarto-Renan/Interruptor/POWER1",
commandTopic="Quarto-Renan/Interruptor/cmnd/POWER1"]
        Type switch : LuzState "Luz State" [ stateTopic="Quarto-Renan/Interruptor/STATE",
transformationPattern="JSONPATH:$.POWER1" ]
        Type switch : Spot  "Spot" [ commandTopic="Quarto-Renan/Interruptor/cmnd/POWER2",
stateTopic="Quarto-Renan/Interruptor/POWER2", on="ON", off="OFF"]
        Type switch : SpotState "Spot State" [ stateTopic="Quarto-Renan/Interruptor/STATE",
transformationPattern="JSONPATH:$.POWER2" ]

        Type string : Fita  "Fita" [ commandTopic="Quarto-Renan/Led" ]

        Type switch : Ventilador  "Ventilador" [ stateTopic="Quarto-Renan/Ventilador/POWER",
commandTopic="Quarto-Renan/Ventilador/cmnd/POWER"]
        Type switch : VentiState  "Ventilador State" [ stateTopic="Quarto-
Renan/Ventilador/STATE", transformationPattern="JSONPATH:$.POWER" ]

        Type contact : Porta "Porta" [stateTopic="Quarto-Renan/Porta"]

        Type number : Sinal "Wifi" [stateTopic="Quarto-Renan/Interruptor/STATE",
transformationPattern="JSONPATH:$.Wifi.RSSI"]

        Type number : Temp "Temperatura" [stateTopic="Quarto-Renan/Sensor/Temperatura"]
        Type number : Umid "Umidade" [stateTopic="Quarto-Renan/Sensor/Umidade"]
    }
}
```

APÊNDICE D – Arquivo de definição de Items: quarto.items

```
DateTime NascerSol "Nascer do Sol [%1$tH:%1$tM]"    <sunrise>   (Astro, gClima)
{channel="astro:sun:local:rise#start"}
DateTime PorSol "Pôr do Sol [%1$tH:%1$tM]"          <sunset>    (Astro, gClima)
{channel="astro:sun:local:set#start"}
String Estacao  "Estação [MAP(est.map):%s]"         <est>       (Astro, gClima)
{channel="astro:sun:local:season#name"}
//
//Lua Items
Number IluminacaoLua  "Iluminação [%d %%]"          <moon>      (Astro, gClima)
{channel="astro:moon:local:phase#illumination"}
String LuaAtual "Lua Atual [MAP(lua.map):%s]"       <lua>       (Astro, gClima)
{channel="astro:moon:local:phase#name"}

Number    Quarto_Cena                "Modo selecionado [MAP(cena.map):%s]"       <cena>
Switch    Quarto_Luz                 "Luz []"                <light>
{channel="mqtt:topic:Broker:QuartoRenan:Luz",
channel="mqtt:topic:Broker:QuartoRenan:LuzState", autoupdate="false"}
Switch    Quarto_Luz2                "Spot []"               <light>
{channel="mqtt:topic:Broker:QuartoRenan:Spot",
channel="mqtt:topic:Broker:QuartoRenan:SpotState", autoupdate="false"}
Switch    Quarto_Vent                "Ventilador []"         <fan>
{channel="mqtt:topic:Broker:QuartoRenan:Ventilador",
channel="mqtt:topic:Broker:QuartoRenan:VentiState"}
Contact   Quarto_Porta               "Porta [MAP(porta.map):%s]" <door>
{channel="mqtt:topic:Broker:QuartoRenan:Porta"}
Number    Quarto_Sinal               "Sinal [%d %%]"         <wifi>
{channel="mqtt:topic:Broker:QuartoRenan:Sinal"}
Number    Quarto_Temperatura         "Temperatura [%.1f ºC]"  <temperatura>
{channel="mqtt:topic:Broker:QuartoRenan:Temp"}
Number    Quarto_Umidade             "Umidade [%d %%]"       <humidity>
{channel="mqtt:topic:Broker:QuartoRenan:Umid"}
Color FitaLedColor "Led Renan" <colorpicker>
String FitaLedString "Led Quarto Renan"        {channel="mqtt:topic:Broker:QuartoRenan:Fita",
autoupdate="false"}
```

APÊNDICE E – Arquivo de definição de Items: midia.items

```
//Items Kodi Renan
Switch KodiRenan_mute "Mudo" <soundvolume_mute> {
channel="kodi:kodi:c0c45743-3dbe-0908-7875-97c12374be15:mute" }
Dimmer KodiRenan_volume "Volume [%d]" <soundvolume> {
channel="kodi:kodi:c0c45743-3dbe-0908-7875-97c12374be15:volume" }
Player KodiRenan_control "Controle" <movecontrol> {
channel="kodi:kodi:c0c45743-3dbe-0908-7875-97c12374be15:control" }
Switch KodiRenan_stop "Início" <stop> { channel="kodi:kodi:c0c45743-
3dbe-0908-7875-97c12374be15:stop" }
String KodiRenan_title "Assistindo [%s]" <play>
String KodiRenan_title2 { channel="kodi:kodi:c0c45743-3dbe-
0908-7875-97c12374be15:title" }
String KodiRenan_showtitle "Show title [%s]" { channel="kodi:kodi:c0c45743-
3dbe-0908-7875-97c12374be15:showtitle" }
String KodiRenan_artist "Artist [%s]" { channel="kodi:kodi:c0c45743-
3dbe-0908-7875-97c12374be15:artist" }
String KodiRenan_playuri "PlayerURI" { channel="kodi:kodi:c0c45743-
3dbe-0908-7875-97c12374be15:playuri" }
String KodiRenan_pvropentv "TV" <mediacontrol> {
channel="kodi:kodi:c0c45743-3dbe-0908-7875-97c12374be15:pvr-open-tv" }
String KodiRenan_notification "Notification" { channel="kodi:kodi:c0c45743-
3dbe-0908-7875-97c12374be15:shownotification" }
String KodiRenan_input "" { channel="kodi:kodi:c0c45743-3dbe-
0908-7875-97c12374be15:input" }
String KodiRenan_inputtext "Inputtext" { channel="kodi:kodi:c0c45743-
3dbe-0908-7875-97c12374be15:inputtext" }
String KodiRenan_systemcommand "Systemcommand" {
channel="kodi:kodi:c0c45743-3dbe-0908-7875-97c12374be15:systemcommand" }
String KodiRenan_mediatype "Mediatype [%s]" {
channel="kodi:kodi:c0c45743-3dbe-0908-7875-97c12374be15:mediatype" }
Image KodiRenan_thumbnail { channel="kodi:kodi:c0c45743-
3dbe-0908-7875-97c12374be15:thumbnail" }
Image KodiRenan_fanart { channel="kodi:kodi:c0c45743-3dbe-
0908-7875-97c12374be15:fanart" }
//Fim Kodi Renan
```

APÊNDICE F – Arquivo de configuração do Sitemap: quarto.sitemap

```
sitemap quarto label="Automação" {

  Frame label="Gerenciamento"{
    Selection item=Quarto_Cena mappings=[1="Filme",2="TV",3="Dormir",4="Nenhum"]
    Default item=Quarto_Sinal

    Text label="Astro" icon="sun"{
    Frame label="Sol"{
      Default item=NascerSol
      Default item=PorSol
      Text    item=Estacao
    }
    Frame label="Lua"{
      Default item=LuaAtual
      Default item=IluminacaoLua
      Default item=Distancia
    }
  }
  }
  Frame label="Iluminação"{
    Default item=Quarto_Luz
    Default item=Quarto_Luz2
    Colorpicker item=FitaLedColor
  }
  Frame label="Controle"{
    Default item=Quarto_Vent
    Default item=Quarto_Porta
    Default item=Quarto_Temperatura
    Default item=Quarto_Umidade
  }
  Frame label="Mídia"{
    Default   item=KodiRenan_title
    Default   item=KodiRenan_control
    Setpoint  item=KodiRenan_volume
    Switch    item=KodiRenan_stop mappings=[ON="OK"]
  }
}
```

APÊNDICE G – Arquivo de definição das Rules: quarto.rules

```
rule "Modo filme"
when
  Item Quarto_Cena changed to 1
then
  Quarto_Luz.sendCommand(OFF)
  Thread::sleep(500)
  Quarto_Luz2.sendCommand(ON)
  FitaLedColor.sendCommand(ON)
end

rule "Apaga as luzes ao dar play"
when
   Item KodiRenan_control changed
then
  if(Quarto_Cena.state == 1){
    if(KodiRenan_control.state == PAUSE)//Acende ao pausar
    {
      Quarto_Luz.sendCommand(ON)
      Thread::sleep(500)//Aguarda meio segundo
      Quarto_Luz2.sendCommand(OFF)
      FitaLedColor.sendCommand(OFF)
    }
    if(KodiRenan_control.state == PLAY)//Apaga ao acender
    {
      Quarto_Luz.sendCommand(OFF)
      Thread::sleep(500)//Aguarda meio segundo
      Quarto_Luz2.sendCommand(ON)
      FitaLedColor.sendCommand(ON)
    }
  }
end

rule "Pausa a reproducao se abrir a porta"
when
    Item Quarto_Porta changed
then
  if(Quarto_Porta.state == OPEN && Quarto_Cena.state == 1){
      Quarto_Luz.sendCommand(ON)
      Quarto_Luz2.sendCommand(OFF)
```

```
    KodiRenan_control.sendCommand(PAUSE)
  }
  if(Quarto_Porta.state == CLOSED && Quarto_Cena.state == 1){
    Quarto_Luz.sendCommand(OFF)
    Quarto_Luz2.sendCommand(ON)
    KodiRenan_control.sendCommand(PLAY)
  }
end
```

APÊNDICE H – Arquivo de definição das Rules: fitaled.rules

```
var HSBType hsbValue
var int redValue
var int greenValue
var int blueValue
var String RGBvalues

rule "Fita led Renan"
when
 Item FitaLedColor changed
then
    hsbValue = FitaLedColor.state as HSBType
    redValue = hsbValue.red.intValue
    greenValue = hsbValue.green.intValue
    blueValue = hsbValue.blue.intValue
    RGBvalues= redValue.toString + ";" + greenValue.toString + ";" + blueValue.toString + ";"
    sendCommand(FitaLedString, RGBvalues)
    logInfo("FitaLedColor", RGBvalues )
end
```

APÊNDICE I – Arquivo de transformação: cena.map

0=Selecione

1=Filme

2=TV

3=Dormir

-=-

NULL=0

APÊNDICE J – Arquivo de transformação: lua.map

```
NULL=
NEW=Lua Nova
WAXING_CRESCENT=Lua Crescente
FIRST_QUARTER=Quarto Crescente
WAXING_GIBBOUS=Lua Gibosa
FULL=Lua Cheia
WANING_GIBBOUS=Lua Balsâmica
THIRD_QUARTER=Quarto Minguante
WANING_CRESCENT=Lua Minguante
```

APÊNDICE K – Arquivo de transformação: porta.map

CLOSED=Fechada

OPEN=Aberta

NULL=null

-=-