UNIVERSIDADE DO ESTADO DO AMAZONAS - UEA

ESCOLA SUPERIOR DE TECNOLOGIA - EST

LUIZ HENRIQUE OLIVEIRA ORTIZ

SISTEMA DE AUTOMAÇÃO RESIDENCIAL COM ÊNFASE EM SEGURANÇA E ECONOMIA ENERGÉTICA

Manaus/AM

2018

LUIZ HENRIQUE OLIVEIRA ORTIZ

SISTEMA DE AUTOMAÇÃO RESIDENCIAL COM ÊNFASE EM SEGURANÇA E ECONOMIA ENERGÉTICA

Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade Estadual do Amazonas, como pré-requisito para a obtenção do título de Engenheiro Eletricista.

Orientador: Dr. Fabio de Sousa Cardoso

Manaus/AM

2018

Universidade do Estado do Amazonas – UEA
Escola Superior de Tecnologia - EST

Reitora:
Cleinaldo de Almeida Costa
Vice-Reitor:
Cleto Cavalcante de Souza Leal
Diretor da Escola Superior de Tecnologia:
Roberto Higino Pereira da Silva
Coordenador do Curso de Engenharia Elétrica
Ingrid Sammyne Gadelha Figueiredo

Banca Avaliadora composta por: *Data da defesa: 14/ 06/ 2018*
Prof. Fábio de Souza Cardoso *(Orientador)*
Prof. Jozias Parente de Oliveira
Prof. José Gomes da Silva

CIP – Catalogação na Publicação

Ortiz, Luiz Henrique Oliveira

Sistema de automação residencial com ênfase em segurança e economia energética / Luiz Henrique Oliveira Ortiz; [orientado por] Fábio de Souza Cardoso. – Manaus: 2018.
81 p.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica). Universidade do Estado do Amazonas, 2018.

1. Automação residencial. 2. *Raspberry Pi*. 3. *Home Assistant*. I. Cardoso, Fábio de Souza.

LUIZ HENRIQUE OLIVEIRA ORTIZ

SISTEMA DE AUTOMAÇÃO RESIDENCIAL COM ÊNFASE EM SEGURANÇA E ECONOMIA ENERGÉTICA

Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade Estadual do Amazonas, como pré-requisito para a obtenção do título de Engenheiro Eletricista.

Nota obtida: _____ (_____)

Aprovada em 14/06/2018.

Área de concentração: Sistemas Embarcados

BANCA EXAMINADORA

_____
Orientador: Fábio de Souza Cardoso, Dr.

_____
Avaliador Jozias Parente de Oliveira, Dr.

_____
Avaliador: José Gomes da Silva, Mr.

Manaus 2018

RESUMO

Tendo em vista a diversidade nas soluções de automação e segurança residencial busca-se com esta pesquisa a implementação de um sistema de automação residencial que visa à melhoria da eficiência energética, a vigilância residencial e que possua baixo custo de investimento. Este sistema possui as funções de: alertar a presença de intrusos e automatizar um condicionador de ar. Caso uma presença seja detectada uma notificação será enviada ao aplicativo instalado em dispositivo *android* ou *IOS* contendo um link de uma *webcam* e botões que possibilitam acionar uma sirene. O condicionador será controlado por agenda de acionamento ou manualmente e possuirá calculo automático de kWh semanais. Para diminuição de custos foi utilizado o *software open source* chamado *Home Assistant,* que pode ser controlado remotamente, instalado no microcontrolador *Raspberry pi*, o número de sensores no circuito foi reduzido à apenas um sensor de presença PIR, um sensor infravermelho, para controle do condicionador, e uma *webcam*, para vigilância do local. Os testes do *software* realizados com o sistema mostraram que o mesmo e suas funcionalidades foram executados atendendo os requisitos de confiabilidade e segurança, os testes de *hardware* do protótipo atendem o objetivo deste trabalho e possuem baixo custo de implementação. O desempenho observado é equiparável a um sistema de automação residencial profissional.

Palavras-chave: automação residencial, *Raspberry Pi*, sistema de vigilância, *Android, IOS, Home Assistant*

ABSTRACT

Considering the diversity of solutions in home automation and security this research seeks the implementation of a home automation system that aims to improve the energy efficiency, make home surveillance and have low investments. This system provides the following functionalities: alert the presence of intruders and automation of air conditioners. When a presence is detected a notification is sent to an application *install*ed on an *android* or *IOS* device containing the link for an *webcam* and push buttons that can activate a siren. The air conditioner is controlled by schedule or manually and will be provided automatic calculation of its kWh weekly. To decrease expenses an *open source software* called Home Assistant, which can be remotely controlled, was embed in the *Raspberry* pi microcontroller, the number of sensors was reduced to only a PIR presence sensor, an infrared sensor, for controlling the air conditioner, and a *webcam*, for local surveillance. The *software* tests conducted on the system reveal that the execution of the *software* and its functions meet the requirements of reliability and security, the *hardware* tests with the prototype comply with this project objectives while having low implementation costs. The observed performance is comparable to a professional home automation system.

Keywords: Home automation, *Raspberry Pi*, surveillance system, *Android, IOS, Home Assistant*

SUMÁRIO

INTRODUÇÃO **8**
1 REFERENCIAL TEÓRICO **10**
1.1 AUTOMAÇÃO RESIDENCIAL 10
1.2 COMUNICAÇÃO SEM FIO 11
1.2.1 Principais caracteristicas na escolha da comunição sem fio **11**
1.2.1.1 Raio de alcance 11
1.2.1.2 *Frequências* 11
1.2.1.3 Consumo energético 11
1.2.1.4 Espalhamento espectral 12
1.2.1.5 *Segurança* 12
1.2.2 Tecnologias de comunicação sem fio de baixo custo **13**
1.2.2.1 *Zigbee* (*xbee*) 13
1.2.2.2 Módulos RF 315MHZ E 433MHZ *15*
1.3 SERVIDOR *WEB* 15
1.4 ANDROID 16
1.5 SENSORES E ATUADORES 17
1.6 SISTEMAS EMBARCADOS 17
1.7 SENSORES DE MOVIMENTO 18
1.7.1 Sensor PIR (Passive Infrared Sensor) **18**
1.7.2 Sensor AIR (Active Infrared Sensor) **18**
2 MÉTODO PROPOSTO **19**
2.1 *SOFTWARE* DE AUTOMAÇÃO *OPEN SOURCE* 19
2.1.1 Home assistant **20**
2.1.2 Home assistant / hassio **21**
2.1.3 Demonstração da linguagem de hassio **21**

2.2 DEFINIÇÃO DOS MATERIAIS EMPREGADOS 27
2.2.1 Microcontrolador 27
2.2.2 Sensores 28
2.2.3 Atuadores 29
2.2.4 Bateria 30
2.3 FLUXOGRAMA DAS FUNÇÕES 30
2.3.1 Fluxograma da função sistema de vigilância 30
2.3.2 Fluxograma da função automação de condicionadores 31
3 IMPLEMENTAÇÃO DO PROJETO 33
3.1 CONFIGURAÇÃO DO *SOFTWARE* 33
3.1.1 IDE (Ambiente de desenvolvimento integrado) 34
3.1.2 Servidor *web* 36
3.2 CONFIGURAÇÃO DA FUNÇÃO SISTEMA DE VIGILÂNCIA 39
3.2.1 Câmera 40
3.2.2 *IOS* 41
3.2.3 *ANDROID* 43
3.3 CONFIGURAÇÃO DA FUNÇÃO AUTOMAÇÃO DE CONDICIONADORES 46
3.4 MONTAGEM DO CIRCUITO 47
3.2.1 Circuito da função de sistema de vigilância 47
3.2.2 Circuito da função de controle de condicionadores 49
4 RESULTADOS OBTIDOS 51
4.1 TESTE DO *SOFTWARE* DA FUNÇÃO SISTEMA DE VIGILÂNCIA 51
4.2 TESTE DO *SOFTWARE* DA FUNÇÃO AUTOMAÇÃO DE CONDICIONADORES 55
4.3 TESTE DO *HARDWARE* DA FUNÇÃO SISTEMA DE VIGILÂNCIA 56
4.4 TESTE DO *HARDWARE* DA FUNÇÃO AUTOMAÇÃO DE CONDICIONADORES 57
4.5 ANÁLISE E COMPARAÇÃO DE CUSTOS 59
CONCLUSÃO 61
REFERÊNCIAS BIBLIOGRÁFICAS 63
FONTES CONSULTADAS 67
APÊNDICE A – CODIGO DO ARQUIVO CONFIGURATION.YAML 69
APÊNDICE B – CODIGO DO ARQUIVO AUTOMATION.YAML 73
APÊNDICE C – CODIGO DO ARQUIVO GROUPS.YAML 80

INTRODUÇÃO

Tecnologias como a da domótica, também conhecida como automação residencial, evoluíram de uma simples ferramenta de automatizar tarefas diárias para uma agregação e interligação de várias tecnologias com propósitos fundamentais para a vida do homem.

Atualmente existem sistemas de automação residencial profissionais que possuem várias funções atraentes ao público como aumento da eficiência energética, melhoria da segurança, automação de tarefas diárias, controle remoto e etc. No entanto, os mesmos possuem alto custo de investimento, o que dificulta a disseminação dessa tecnologia na sociedade (IHOUSE, 2018).

Partindo dessa ideia, o presente trabalho tem como tema desenvolver um sistema de automação residencial de baixo custo para automação de condicionadores de ar e que funcione como sistema de vigilância.

Sistemas que utilizam microcontroladores como fonte de processamento possuem baixo consumo de energia, na ordem de 4W, e portanto há possibilidade de conexão de uma bateria de reserva que permite o funcionamento de algumas funções da central mesmo na falta de energia elétrica, possibilitando um sistema de vigilância que funcione integralmente durante oscilações e falta de energia elétrica.

Partindo-se da hipótese de que é possível o desenvolvimento de um sistema de automação residencial que possua custo total inferior ao de mercado, tendo ele base em tecnologias de código aberto (do inglês, *open source*) e plataforma de prototipagem adequada, o presente trabalho tem como objetivo a pesquisa e implementação de um sistema de automação *open source* que possibilite controle à distância por meio de aplicativo *android* ou *IOS*, que utilize uma tecnologia de comunicação sem fio e seja capaz de incorporar as funções pretendidas com eficiência, sendo elas: função de sistema de vigilância e função de automatização de

condicionadores de ar. Após projetado, o sistema teve seus componentes eletrônicos interligados e suas funções testadas para confirmação da hipótese.

Esta pesquisa se justifica por fornecer uma documentação implementada de um dos sistemas de automação *open source* mais empregados atualmente, implementação essa de um sistema com programação multifuncional, com baixo custo de implementação, de fácil controle por meio de aplicativo *android* ou *IOS* utilizado em qualquer rede.

Este trabalho está dividido em quatro seções além das referências, que serão citadas a seguir:

Primeiramente a seção de Referencial Teórico que descreve as tecnologias avaliadas para implementação nesse projeto.

A segunda seção, chamada Método Proposto, descreve os materiais e tecnologias utilizados no projeto assim como familiariza o leitor com a programação dos *softwares* empregados.

A terceira seção, Implementação do Projeto, consiste na apresentação dos passos que devem ser seguidos para configuração e programação dos *softwares*, cujos códigos de programação se encontram nos apêndices A, B e C, montagem dos circuito e interligação do *hardware*.

A quarta seção, Resultados Obtidos, descreve os resultados dos testes das funções programadas verificando itens como tempo de atraso para ativação dos atuadores, estabilidade dos sensores de *GPS* e presença, eficiência do sistema sem energia da rede elétrica e estabilidade do aplicativo *android.*

1 REFERENCIAL TEÓRICO

1.1 AUTOMAÇÃO RESIDENCIAL

Automação residencial é o conjunto de serviços proporcionados por sistemas tecnológicos integrados como o melhor meio de satisfazer as necessidades básicas de segurança, comunicação, gestão energética e conforto de uma habitação. Nesse contexto, costumamos achar mais adequado o termo "Domótica", largamente empregado na Europa, pois é mais abrangente (CARAUDIORJ, 2018). Segundo a *Asociación Española de Domótica e Inmótica* (CEDOM, 2018) pode-se definir Domótica, como:

> Domótica é a automatização e o controle aplicados à residência. Esta automatização e controle se realizam mediante o uso de equipamentos que dispõem de capacidade para se comunicar interativamente entre eles e com capacidade de seguir as instruções de um programa previamente estabelecido pelo usuário da residência e com possibilidades de alterações conforme seus interesses. Em consequência, a domótica permite maior qualidade de vida, reduz o trabalho doméstico, aumenta o bem-estar e a segurança, racionaliza o consumo de energia e, além disso, sua evolução permite oferecer continuamente novas aplicações.

Conceitualmente o termo domótica é utilizado como uma definição mais abrangente de *Home Automation* por envolver, por exemplo, sistemas de sonorização. Nesse trabalho foi utilizado o termo *Home Automation* para representar quaisquer serviços proporcionados por sistemas tecnológicos integrados para satisfazer as necessidades básicas de segurança, comunicação, gestão energética e conforto de uma habitação. Ao unir *Home Automation* com loT (*Internet of Things*) e sistemas embarcados vemos que esses três conceitos são complementares (ALMEIDA, 2016).

Conforme publicação da gazeta do povo (BUBNIAK, 2013), por Taiana Bubniak, um kit simples de Automação residencial da fabricante Schutz Automação®, custa R$ 3.600,00. O pacote inclui a automação de dois circuitos de iluminação, *home theater* e o controle do ar-condicionado.

Na *Autoprojects*®, um conjunto de soluções compactas para uma sala de *home* cinema, integrando iluminação, TV, *Blu-ray* player, cortinas motorizadas e ar-condicionado, com telas de controle por *smartphones* e *tablets*, custa R$ 10 mil (BUBNIAK, 2013).

Na AutomacaoResidencialBrasil® os serviços de automação tem custos a

partir de R$ 3.300,00 (AUTO, 2018). A empresa *Ihouse* de automação fornece automação de sala de estar por R$ 4.590,00 (IHOUSE, 2018).

Pode-se observar que os preços de sistemas de automação residencial se mantiveram constantes ao passar dos anos devido principalmente à adição de novas tecnologias e fornecimento de apoio técnico 24 horas, no entanto não existe comercialização a preço popular da tecnologia genérica.

1.2 COMUNICAÇÃO SEM FIO

No mercado atual, diversas são as opções de sistemas de Automação Residencial sem Fio. Um estudo aprofundado das tecnologias principais é essencial.

1.2.1 Principais características na escolha das redes sem fio

1.2.1.1 *Raio de Alcance*

Determinar o raio de transmissão de uma rede sem fio é um fator determinante na definição do modo de operação de cada aplicação, deixando-a largamente maleável em razão de tal mobilidade de modo que seus usuários possuam um tipo de limitação, que é a sua área de abrangência ou raio de alcance. Em se tratando do protocolo RF, seu alcance é similar ao de redes com capacidades mínimas de 10m e a máxima de 100 metros (TELECO, 2017).

1.2.1.2 *Frequências*

A frequência ou espectro eletromagnético é considerado um dos recursos de maior valia visto que não é exaurível mesmo que a demanda de uso seja cada vez maior e que esta seja limitada por região. As frequências são adquiridas juntas a órgãos governamentais já que o espectro é patrimônio público (TELECO, 2017).

Como alternativa, é grande o número de países que disponibiliza frequências gratuitas e não licenciadas para aplicações de uso nas indústrias, medicinas e para fins científicos chamadas de ISM (*Industrial Scientifical and Medical*). Estas faixas de ISM podem variar de acordo com o país (LUDWIG, 2015).

1.2.1.3 *Consumo Energético*

Um dos principais problemas das redes sem fio é o consumo energético. Esforços dos fabricantes de equipamentos seguem no sentido de desenvolver equipamentos com cada vez menos consumo energético.

O baixo consumo energético é vital no desenvolvimento de qualquer projeto e deve possuir reduzida complexidade de tal forma que possa permitir o uso de baterias como fonte de alimentação. Além disso, o sistema deve possuir uma lógica de transmissão eficiente que possibilite o menor esforço na transmissão, consequentemente, gerando um baixo consumo energético (LUDWIG, 2015).

1.2.1.4 *Espalhamento Espectral*

Espalhamento espectral é uma técnica de modulação na qual a energia média do sinal transmitido é espalhada de forma muito maior que a banda mínima necessária para transmitir a informação. Deste modo, a energia do sinal transmitido ocupa uma banda bem maior do que a do dado transmitido.

Trata-se de uma técnica de codificação para transmissão digital de sinais desenvolvida originalmente para utilização militar, com o objetivo de transformar a informação transmitida em um sinal semelhante a um ruído, evitando monitoração pelos adversários (TELECO, 2017).

Devido ao fato de que as faixas utilizadas neste modelo de transmissão apresentam uma larga quantidade de sinais interferentes, o uso desta lógica se justifica em aplicações de uso não licenciado do espectro, visto que a banda de frequências disponível é dividida em canais independentes que no decorrer do tempo, tem sua frequência de transmissão dos dados alterada.

Comumente, se comenta sobre o espalhamento espectral com salto em frequência (FHSS), em situações onde as frequências de transmissão são alteradas de maneira aleatória, através do uso de um *software* específico a este fim. Ou, espalhamento espectral com sequência direta (DSSS), nos casos em que os dados são enviados multiplicados por um sinal codificador, de maneira semelhante a tecnologia de múltiplo acesso por divisão de código (CDMA).

Seja qual for o método aplicado, percebem-se resultados positivos no que diz respeito à redução dos efeitos causados por sinais externos (LUDWIG, 2015).

1.2.1.5 *Segurança*

Com uma demanda cada vez mais crescente de usuários a necessidade de cuidados com a segurança é eminente, sendo assim o estudo de técnicas de chave que possibilitem que o acesso às interconexões seja restrito apenas aos usuários da rede são empregadas.

Técnicas como WEP (*Wired Equivalent Privacy*) e WPA (*Acesso Protegido Wi-Fi*), são cada vez mais usados em conjuntos a *firewalls* em dispositivos móveis ou não para garantir a segurança e privacidade nas trocas de informações (CONCEIÇÃO JUNIOR, 2012).

1.2.2 Tecnologias de Comunicação Sem Fio

1.2.2.1 *ZigBee (Xbee)*

Este protocolo foi projetado para permitir comunicação sem fio confiável, com baixo consumo de energia e baixas taxas de transmissão para aplicações de monitoramento e controle. Para implementar as camadas MAC (*Medium Access Control*) e PHY (*Physical Layer*) o *ZigBee* utiliza a definição 802.15.4 do IEEE (RFC 4441), que opera em bandas de frequência livres. A tecnologia *ZigBee* foi pensada para interligar pequenas unidades de coleta de dados e controle recorrendo a sinais de radiofrequência não licenciados.

A tecnologia utilizada é comparável às redes *Wi-Fi* e *Bluetooth* e diferencia-se destas por desenvolver menor consumo, por um alcance reduzido (cerca de 100 metros) e a comunicação entre duas unidades poder ser repetida sucessivamente pelas unidades existentes na rede até atingir o destino final. Todos os pontos da rede podem funcionar como retransmissores de informação. Uma malha (*Mesh*) de unidades *ZigBee* pode realizar-se numa extensão doméstica sem necessidade de utilizar ligações elétricas entre elas (LUDWIG, 2015).

Os fornecedores de chips *ZigBee* tipicamente vendem rádios integrados e microcontroladores com memória flash entre 60 KB e 256 KB. O *ZigBee* opera nas faixas de rádio industriais, científicas e médicas (ISM), 868 MHz na Europa, 915 MHz nos EUA e Austrália, e de 2,4 GHz na maioria das jurisdições em todo o mundo. Suas taxas de transmissão de dados variam de 20 a 900 *kilobits* por

segundo. A camada de rede *ZigBee* suporta nativamente a topologia estrela e árvore em arquiteturas de malha genérica. Cada rede deve ter um dispositivo coordenador, encarregado de sua criação, o controle de seus parâmetros e manutenção básica.

O *ZigBee* pode ir do modo sono para o modo ativo em 30ms ou menos, a latência é baixa e dispositivos podem ser ágeis, particularmente em comparação com os atrasos do *Bluetoothwake-up*, que é tipicamente em torno de três segundos. Como os nós *ZigBee* podem dormir a maior parte do tempo, o consumo de potência média pode ser reduzida, resultando em longa duração da bateria.

Existem três tipos diferentes de dispositivos:

a) *ZigBee*: *ZigBee* coordenador (ZC): O dispositivo mais completo, o coordenador faz a raiz da árvore da rede e pode superar a outras redes. Há exatamente um coordenador *ZigBee* em cada rede, uma vez que é o dispositivo que iniciou a rede originalmente. Ele é capaz de armazenar informações sobre a rede, inclusive atuando como o Centro de Fidedignidade e repositório de chaves de segurança;

b) *ZigBee Router* (ZR): Assim como executar uma função do aplicativo, um roteador pode funcionar como um roteador intermediário, transmissão de dados de outros dispositivos;

c) *ZigBee* dispositivo final (ZED): Contém a funcionalidade apenas o suficiente para falar com o nó pai (ou coordenador ou um roteador), ele não pode transmitir dados de outros dispositivos. Esta relação permite que o nó a ser adormecido uma quantidade significativa do tempo dando assim a vida da bateria longa. Um ZED requer menor quantidade de memória, e, portanto, pode ser menos caro de fabricar do que um ZR ou ZC (LUDWIG, 2015).

Uma vantagem do Protocolo *ZigBee* comparado a outras tecnologias é a baixa taxa de transmissão da rede. Isso por que, devido à baixa capacidade de transmissão a rede pode dispor de um menor consumo energético. Tal consumo pode ser tão reduzido que permite o uso contínuo por até 6 meses com alimentação exclusiva de baterias do tipo AA.

Outra grande vantagem deste padrão é a capacidade de, através do uso de topologias de rede (mista, malha e árvore), este protocolo fornece a capacidade de uso de 65000 nós simultâneos.

A desvantagem do *ZigBee* são as baixas taxas de transferência o que mesmo para aplicações simplórias o inviabiliza em certas ações que precisam de

uma taxa de transferência maior as que ele possui, como no caso da interconexão entre dispositivos (CONCEIÇÃO JUNIOR, 2012).

Seu custo de aquisição nos módulos mais simples de comunicação pode variar de R$ 100,00 a R$ 500,00, por módulo.

1.2.2.2 Módulos RF 315Mhz e 433Mhz

Um módulo de Rádio Frequência (RF) é um (normalmente) pequeno dispositivo eletrônico usado para transmitir e/ou receber sinais de rádio entre dois dispositivos. Circuitos de rádio estão sujeitos a limites de emissões irradiadas normalmente, e exigem testes de conformidade e certificação por uma Organização de Normalização: como ETSI ou a Comissão Federal de Comunicações dos EUA (FCC). Por estas razões, os projetistas que desejam utilizar comunicação por RF em um circuito acabam por utilizar um módulo RF pré-fabricado em vez de tentar um design discreto, economizando, dessa maneira, tempo e dinheiro no desenvolvimento.

Módulos de RF são mais frequentemente utilizados em produtos de volume médio e baixo para aplicações de consumo: como abridores da porta da garagem, sistemas de alarme sem fio, controles remotos industriais, aplicações de sensores inteligentes e sistema de automação residencial sem fio. Eles são por vezes usados para substituir projetos de comunicação infravermelho mais velhos, pela vantagem de não necessitar de operação com linha de visão. Várias são as frequências utilizadas para a comunicação dos módulos de RF comercialmente disponíveis, incluindo as de uso Industrial, Bandas de Rádio Científicas e Médicas (ISM): tais como 433.92 MHz, 315 MHz, 868 MHz, 915 MHz, 2400 MHz. Estas frequências são usadas por causa dos regulamentos nacionais e internacionais que regem as faixas de frequência de uso livre (LUDWIG, 2015).

1.3 SERVIDOR *WEB*

Toda aplicação *web* deve ser hospedada em um computador que pode ser acessado por todos os clientes. Para que todo este processo ocorra de forma estável e com segurança, faz-se necessária uma configuração do computador capaz de interpretar e responder todas as solicitações vindas dos clientes. Este

computador cria um servidor *web* e deve ser tal que a estrutura possua containers capazes de hospedar e executar diversos scripts e estruturas completas de uma aplicação que possa contar com acessos a um gerenciador de banco de dados (W3TECHS, 2017).

Atualmente há diversas aplicações disponíveis que transformam simples computadores em servidores *Web*. A escolha de qual aplicação utilizar está ligada normalmente a linguagem padrão na qual o sistema foi desenvolvido e recursos de *hardware* disponíveis. Dentre as possibilidades, destaca-se o *Apache Tomcat* que é uma implementação de *software* de código aberto de tecnologia que executa códigos escritos na linguagem Java.

1.4 *ANDROID*

O *Android* é um sistema operacional desenvolvido pela empresa Google. Sua primeira versão comercial foi lançada no ano de 2008, e, ao longo dos anos, foi aprimorado e passou a ser empregado em diversas funcionalidades.

O *Android* ganhou muito espaço com o surgimento dos *smartphones* e *tablets*. Diversas das grandes montadoras de dispositivos móveis optaram em lançar seus aparelhos com o sistema operacional *Android*, tornando-o sistema operacional móvel mais difundido e utilizado no mundo.

Este sistema operacional trabalha com uma máquina virtual chamada Dalvik *Virtual Machine*, que compila os programas desenvolvidos em Java para dispositivos móveis. Todas as aplicações podem ser executadas independentemente do modelo e fabricante do aparelho, exceto os casos onde a limitação é de *hardware*.

Para o desenvolvimento de aplicativos, faz-se necessária a instalação de uma IDE. Logo, é necessário instalar e configurar um *plugin* chamado de ADT (*Android Development Tools*), que foi desenvolvido para proporcionar um ambiente integrado para o desenvolvimento de aplicativos para *Android*. É uma ferramenta completa, que amplia os recursos da IDE Eclipse, na qual pode-se criar rapidamente as aplicações com recursos de interface, realizar a depuração em um simulador que pode ser selecionado de acordo com o ambiente do dispositivo móvel, e, por fim, possibilitar a criação do arquivo final para a distribuição (BEGHINI, 2013).

1.5 SENSORES E ATUADORES

Sensores são dispositivos capazes de mensurar grandezas físicas do meio e transformar em um sinal elétrico que podem então ser processados e interpretados por outros equipamentos como por exemplo microcontroladores e computadores. Pode se dizer que são componentes capazes de permitir um equipamento eletrônico a interagir e sentir o mundo.

Segundo Borges e Dores (2010), sensores que operam de forma direta, ou seja, transformando uma forma de energia em outra são chamados de transdutores. Já sensores que atuam de forma indireta alteram suas propriedades sob ação da grandeza medida de forma que essa alteração ocorra de forma proporcional. Um bom exemplo são os fotoresistores que alteram sua resistência de acordo com a quantidade de luz do ambiente.

Os atuadores são considerados também transdutores mas fazem o caminho contrário aos sensores, eles transformam sinais elétricos em grandezas físicas sendo capazes de realizar ações que modificam o ambiente em que estão inseridos.

Brugnari e Maestrelli (2010) dizem que atuadores atendem a comandos que podem ser manuais ou automáticos, ou seja, qualquer elemento que realize um comando recebido de outro dispositivo, com base em uma entrada ou critério a ser seguido. Um bom exemplo de atuador é um relé que funciona com pequenas correntes, mas é capaz de ativar ou desativar circuitos externos que possuem correntes elevadas.

1.6 SISTEMAS EMBARCADOS

Os sistemas embarcados podem ser definidos como sistemas de processamento digital que possuem seu *software* de controle (denominado *firmware*) armazenado em uma memória presente no circuito eletrônico. São dedicados a uma tarefa específica e interagem com o ambiente por meio de sensores e atuadores, sendo utilizados em aviões, carros, telefones celulares, calculadoras, eletrodomésticos, equipamentos médicos etc. De forma geral, são empregados na maioria dos equipamentos eletrônicos atuais (SCHNEIDER e SOUZA, 2010).

Esses sistemas possuem características que os distinguem dos outros sistemas operacionais convencionais, são elas:

- Funcionalidade única: são desenvolvidos para atender a uma funcionalidade específica enquanto sistemas operacionais convencionais podem ter diversas funcionalidades.
- Restrições de projeto mais rígidas: sistemas embarcados são geralmente Desenvolvidos utilizando microcontroladores que possuem recursos muito limitados em comparação com os microprocessadores utilizados por sistemas operacionais Convencionais.
- Sistemas reativos em tempo real: sistemas embarcados devem reagir a mudanças no ambiente e fornecer resultados em tempo real (LONGO, 2015).

1.7 SENSOR DE MOVIMENTO

1.7.1 Sensor PIR (*Passive Infrared Sensor*)

Esse tipo de sensor não emite luz infravermelha, por isso caracterizado como passivo. Pelo contrário, eles fazem uma leitura nas mudanças do infravermelho do ambiente. A radiação infravermelha existe no espectro eletromagnético e são invisíveis ao ser humano, objetos que geram calor também geram radiação infravermelha. Quando o sensor é energizado ele faz uma leitura do infravermelho que esta retornando do ambiente sem movimento (BEGHINI, 2013).

1.7.2 Sensor AIR (*Active Infrared Sensor*)

Esses sensores trabalham em pares, o transmissor emite ondas de infravermelho (um tipo de luz que o ser humano não enxerga) para o receptor. Quando alguém ou alguma coisa corta o feixe de luz o receptor detecta “utilizado nas portas dos elevadores e nos portões automáticos para não fechar a porta ou o portão quando houver alguém ou alguma coisa em frente” (BEGHINI, 2013).

2 MÉTODO PROPOSTO

Quanto à natureza o presente trabalho é uma pesquisa aplicada, cujo objetivo foi a realização de uma pesquisa exploratória sobre o material bibliográfico e de laboratório adquiridos sobre o assunto. Os procedimentos técnicos a serem utilizados foram os de pesquisa bibliográfica e comparativa. Como método de abordagem, foi utilizado o hipotético-dedutivo e a elaboração seguiu o método de procedimento monográfico. A coleta de dados foi feita através da observação direta intensiva e documentação indireta, sendo estes dados quantitativos.

Inicialmente, foram realizadas pesquisas bibliográficas na área de automação industrial, comunicações sem fio, sistemas de segurança, eficiência energética, acionamento de reles, servidor *web* e aplicativos *android.*

A seguir estão descritos os subcapítulos do desenvolvimento do projeto:

1. *SOFTWARE* DE AUTOMAÇÃO *OPEN SOURCE* – Apresentação e documentação da linguagem utilizada no *software open source* definido.
2. DEFINIÇÃO DOS MATERIAIS EMPREGADOS – Características das tecnologia aplicadas no projeto de acordo com custo benefício e atendimento dos requisitos do sistema.
3. FLUXOGRAMA DAS FUNÇÕES – Definição dos diagramas de bloco das funções desempenhadas pelo sistema.

2.1 *SOFTWARE* DE AUTOMAÇÃO *OPEN SOURCE*

Uma pesquisa foi realizada entre os softwares de automação *open source* para a escolha do que apresente menores requerimentos de hardware, que tenha configuração simplificada e que possua uma grande comunidade, Figura 2.1. A comunidade é de grande importância para um *software open source* pois quanto maior o numero de contribuidores ao domínio mais rápido será o crescimento do *software*, maior o numero de exemplos de programas e maior o numero de novos programas não oficiais.

Figura 2.1 – Trafico global nos domínios dos maiores sistemas *open source*

DOMOTICZ

OPENHAB

HOME ASSISTANT

Fonte: ALEXA, 2018.

2.1.1 *Home Assistant*

Home Assistant foi selecionado para o projeto em questão por possuir as seguintes qualidades superiores:

- O *software* é atualizado regularmente durante o período de quatro anos de sua criação, tornando-se um dos *softwares* com maior crescimento no número de funcionalidade e compatibilidade com outros programas contendo até o presente dia 1111 componentes virtuais que podem adicionar funções ou interligar outros *softwares* e *hardwares* conhecidos ao sistema;
- É capaz de realizar automação de forma simplificada por usar codificações de dados para programação, o que permite criação de funções pela interface gráfica apenas pela seleção das variáveis disponíveis, além de contar com uma vasta comunidade de auxilio que fornece códigos de exemplo;

- É compatível com qualquer plataforma microcontroladora capaz de rodar Python;
- Partes do *software* podem ser instaladas facilmente em módulos se necessário por meio de *Add-on*s;

Existem no momento três maneiras de instalar *Home Assistant* em uma placa micro controladora (HASSIO, 2018):

- *Hass.io*: É o mais novo método de instalação que possibilita fácil instalação e atualização dos componentes. É o método sugerido no site e pode ser instalado *Raspberry* Pi ou Intel NUC. O único método com a funcionalidade de instalar add-ons, não necessita uso de comandos em *cmd*;
- *Python Virtual Env*: Permite a instalação em qualquer *hardware* que execute Python por meio da criação de ambiente virtual, necessita uso de comandos em *cmd*;
- *Hassbian*: É um SO modificado com base no Raspbian onde o *Home Assisant* vem pré instalado, possibilita o funcionamento separado tanto do SO como do sistema de automação, utiliza alguns comandos em *cmd*.

2.1.2 *Home Assistant* / *Hass.io*

Além das funções já citadas essa variação do *Home Assistant* proporciona atualizações automáticas, automação rápida e simplificada por meio de interface gráfica, alta compatibilidade e aquisição de *software* por meio de *Add-on*s e as facilidades de fazer uma cópia completa do estado do sistema, que pode ser instalada em outra placa *Raspberry* de configuração igual ou superior sem necessidade de reconfigurar o sistema.

Home Assistant Hass.io foi lançado pela licença *Apache 2.0*, garantindo que o código é seguro.

2.1.3 Demonstração da linguagem de *Hass.io*

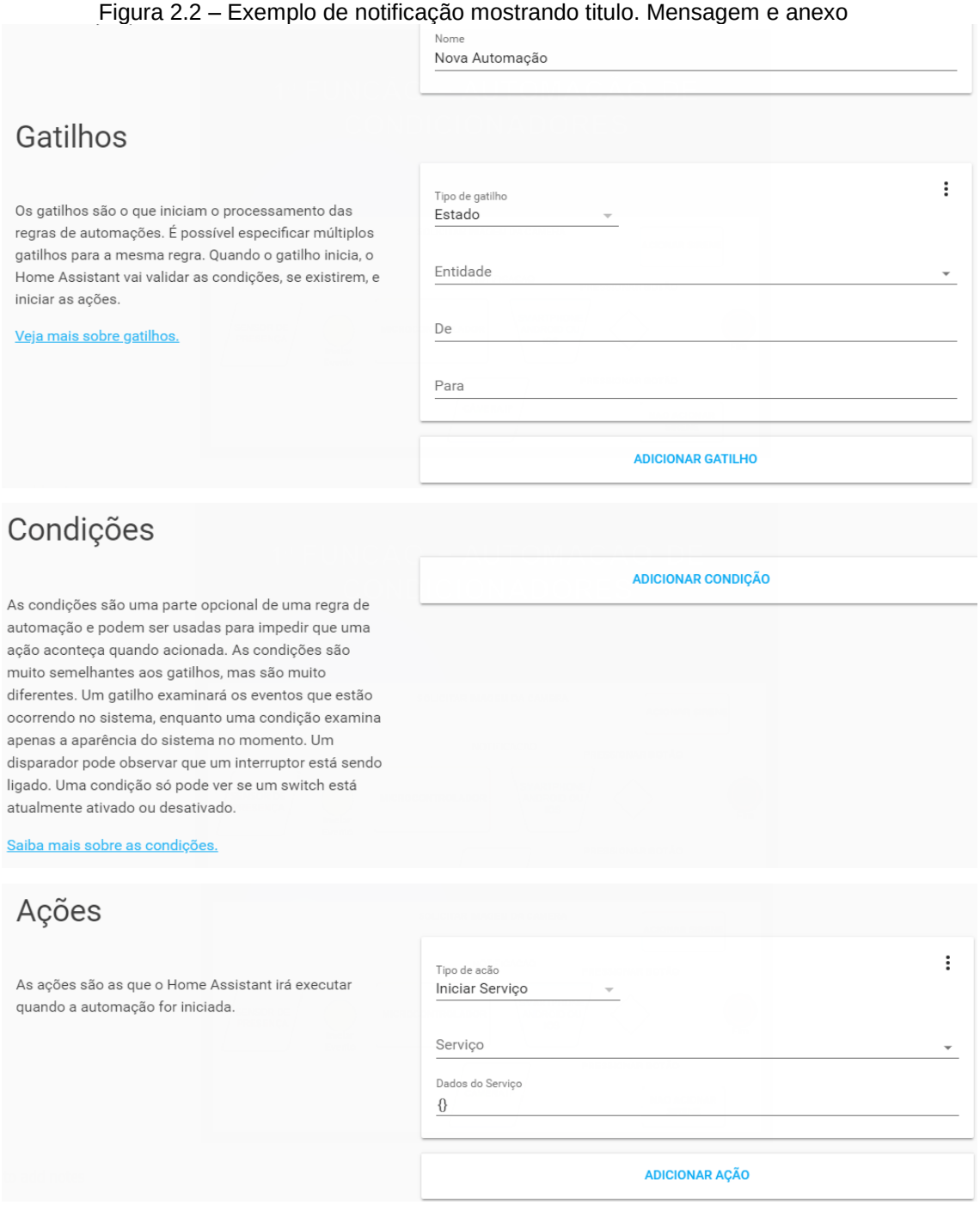
O sistema *'Home Assistant*' funciona com a linguagem *'python 3'* no entanto sua programação é realizada pelo método de codificação de dados YAML (YAML *Ain't Markup Language*) que provem do JSON (*JavaScript Object Notation*) e XML

(*Extensible Markup Language*) que é de fácil manuseio, os dois arquivos a seguir são utilizados para configuração do sistema:

- *Configuration.yaml*: Arquivo principal que contém todas as componentes instaladas e configurações do sistema, cada alteração necessita reiniciar todo sistema para surtir efeito.
- *Automation.yaml*: Arquivo inicialmente vazio onde são adicionadas as funções que o sistema deve desempenhar. Após cada alteração uma parte do sistema deve ser reiniciada pressionando o botão '*reload automations'* localizado no painel principal chamado '*Home Assistant'* no canto superior esquerdo da interface principal na aba *'Configuration'* dentro de 'General'.

As funções do sistema, automações, podem ser configuradas sem fazer o uso de linhas de código em YAML com uso da interface de usuário acessando o painel principal no canto superior esquerdo da interface principal na aba 'Configuration' dentro de 'Automation', figura 2.2, cuja documentação se encontra no site do sistema (EDITOR, 2018), no entanto toda programação apresentada no presente trabalho é realizada em linhas de código YAML, cuja documentação (YAML, 2018) e exemplos (EXEMPLOS, 2018) se encontram no site.

Figura 2.2 – Exemplo de notificação mostrando titulo. Mensagem e anexo



Fonte: Próprio autor, 2018.

A arquitetura consiste nos seguintes itens:

- Domínio: Tipo que uma entidade pode ter, existe uma lista definida de domínios para organizar as entidades, por exemplo ‘*light’* ou ‘*sensor’*;
- Entidades: São representações de sensores, atuadores ou qualquer variável que possa ser utilizada em automações, são representadas pelo seu domínio seguido do nome escolhido, exemplo’ *sensor.sensor_luiz’*;

- Estado: Estado que a entidade se encontra, dentro dos estados possíveis de cada uma, exemplo ‘*on’*;
- Atributos: Valores adicionais, além do estado, que uma entidade pode ter, exemplo ‘*hidden: false*’;
- Serviços: Comando que gera uma ação, cada domínio tem seus possíveis serviços, consiste do domínio seguido do nome do serviço, exemplo ‘*light.turn_on’*;
- Componentes: define a lógica de uma funcionalidade, por exemplo ‘*notify*’ envia notificações para outros *software*s;
- Plataforma: Realizam conexão entre um componente e um *software* ou *hardware* especifico, por exemplo ‘*pushbullet’* liga o componente ‘*notify’* ao site pushbullet.com;
- *Templates*: Método de programação mais avançado em JSON que oferece maior número de variáveis de acesso ao sistema e mais opções de lógica de programação.

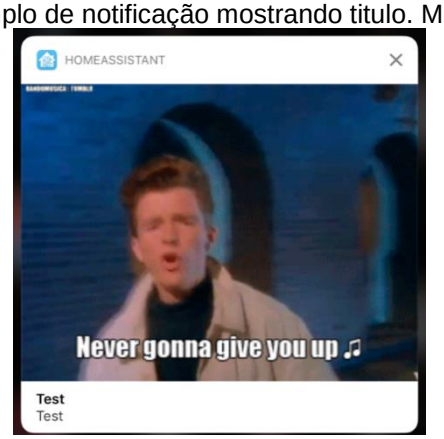
A seguir se encontra uma demonstração da linguagem ‘*Yaml*’, mostrada com intuito de melhor documentar a configuração do sistema.

Para gerar qualquer ação um código chamado ‘*automação*’ deve ser copiado para o final do arquivo chamado ‘automation.yaml’, para a criação de uma automação qualquer seu código deve conter um *‘Trigger’*, uma *‘Condition’* e uma ‘*Action*’, onde a *‘Condition’* é optativa, cada item tem a seguinte função:

- *Trigger:* Gatilho que irá iniciar a ação, ‘*Action’,* podem haver vários gatilhos em uma automação, cada gatilho necessita que uma plataforma seja declarada;
- *Condition:* Requisito adicional ao *Trigger* que deverá ser cumprido para iniciar a ação, também necessita que uma plataforma seja declarada, mas possui variáveis diferentes do *Trigger* para cada plataforma*;*
- *Action:* Ação que será realizada, necessita que um ou mais serviços sejam declarados, uma lista de serviços disponíveis pra cada domínio pode ser encontrada no painel principal em ferramentas de desenvolvedores.

A seguir um exemplo de notificação simples enviada para o aplicativo *IOS*, Figura 2.3.

Figura 2.3 – Exemplo de notificação mostrando titulo. Mensagem e anexo



Fonte: HOMEASSISTANT, 2018.

A notificação acima pode ser reproduzida pelo código na figura 2.4:

Figura 2.4 – Codificação de dados em YAML para notificação

```yaml
- alias: 'Notificar app'
  trigger:
    platform: state
    entity_id: switch.Sirene
    from: 'off'
    to: 'on'
  action:
    service: notify.ios_iphonetcc
    data:
      title: "Test"
      message: "Test"
      data:
        subtitle: ""
        push:
          sound: "US-EN-Alexa-Motion-At-Front-Door.wav"
        attachment:
          url: https://67.media.tumblr.com/tumblr_nfn3ztLjxk1tq4of6o1_400.gif
          content-type: gif
```

Fonte: Próprio autor, 2018.

A seguir breve explicação dos itens em destaque:

- *alias*: nome da automação, aparecera na *interface* de usuário;
- *entity_id*: tipo da entidade seguido de seu nome, para verificação de estado pela plataforma, no caso do estado ‘off’ para ‘on’;
- *service:* serviços do domínio *notify* de nome ios_x, iphonetcc deve ser substituído pelo nome definido no aplicativo;
- *title*: título da notificação, aparece no *Apple Watch* e *IOS* a partir do 10;
- *message*: mensagem da notificação;
- *subtitle*: legenda da notificação;
- *sound*: som da notificação escolhido da lista disponível em (HOMEASSISTANT, 2018);

- *attachment*: url qualquer, podendo ser de imagem, vídeo ou áudio
- *URL*: endereço do anexo na internet;
- *content_type*: o tipo do arquivo.

Os anexos podem ter as seguintes extensões no servidor IOS padrão:

- Anexo de áudio: tamanho máximo 5 MB, pode ter extensão AIFF, WAV, MP3 ou MPEG4 Áudio;
- Anexo de imagem: tamanho máximo 10 MB, pode ter extinção JPEG, GIF ou PNG;
- Anexo de vídeo: tamanho máximo 50 MB, pode ter extensão MPEG, MPEG2, MPEG4 ou AVI.

É possível adicionar um vídeo em tempo real, *stream*, em uma notificação com o código contido na figura 2.5 desde que a câmera suporte vídeos em MJPEG.

Figura 2.5 – Codificação em YAML para criação de notificação com vídeo

```
57  - alias: 'Movimentodetectado'
58    trigger:
59      platform: state
60      entity_id: binary_sensor.Presenca
61      from: 'off'
62      to: 'on'
63    action:
64      service: notify.ios_iphonetcc
65      data:
66        message: "sensor acionado"
67        data:
68          attachment:
69            content-type: jpeg
70          push:
71            category: camera
72          entity_id: camera.demo_camera
```

Fonte: Próprio autor, 2018.

A seguir breve explicação dos itens em destaque:

- entity_id linha 60: Entidade presença do domínio sensor binário;
- entity_id linha 72: tipo de entidade seguido de seu nome, no caso foi utilizado uma câmera chamada demo_camera.

2.2 DEFINIÇÃO DOS MATERIAIS EMPREGADOS

2.2.1 Microcontrolador

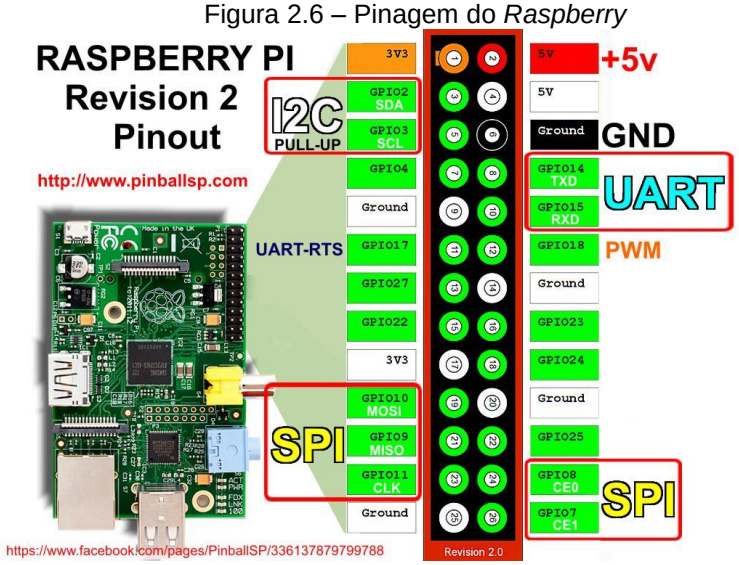
A placa microcontroladora definida para o sistema é o *Raspberry Pi Model B Rev* 2.0 512MB, é um *hardware* potente e amplamente utilizado por um número crescente de tecnologias, apesar de o modelo em questão ter sido lançado em 2012 a placa demonstrou desempenho mais que necessário para acomodar o robusto sistema baseado em Linux. A tabela 2.1 informa as especificações e a figura 2.6 sua pinagem.

Tabela 2.1 – Especificações do *Raspberry* empregado

SoC	Broadcom BCM2835 (CPU, GPU, DSP, SDRAM, and Single USB Port)
CPU	700 MHz ARM1176JZF-S Core (ARM11 Family)
GPU	Broadcom VideoCore IV, OpenGL ES 2.0 (24 GFLOPS)
Memory (SDRAM)	512 MB (Shared with GPU)
USB 2.0 Ports	2 (Via the built in integrated 3-Port USB Hub)
Video Outputs	HDMI (Rev 1.3 & 1.4), Composite RCA (PAL and NTSC)
Audio Outputs	3.5mm Jack, HDMI
Onboard Storage	SD / MMC / SDIO Card Slot (3,3V Card Power Support Only)
Onboard Network	10/100 Ethernet (8P8C) USB adapter on the third port of the USB hub
Low-level Peripherals	8 × GPIO, UART, I²C bus, SPI bus with two chip selects, I²S Audio +3.3 V, +5 V, ground
Power Ratings	700 mA (3.5 W)
Power Source	5 Volt via MicroUSB or GPIO Header
Size	85.60 mm × 53.98 mm (3.370 in × 2.125 in)
Weight	45 g (1.6 oz)
Operating Systems	Arch Linux ARM, Debian Linux, Fedora, FreeBSD, Plan 9, Raspbian OS, RISC OS, Slackware Linux

Fonte: BOXTECH, 2018.

Figura 2.6 – Pinagem do *Raspberry*



Fonte: CHRISATECH, 2018.

O microcontrolador em questão não é mais vendido, portanto a versão *RASPBERRY PI 2 MODEL* B de fevereiro de 2015 é a opção com menor custo atualmente, aproximadamente 170 reais.

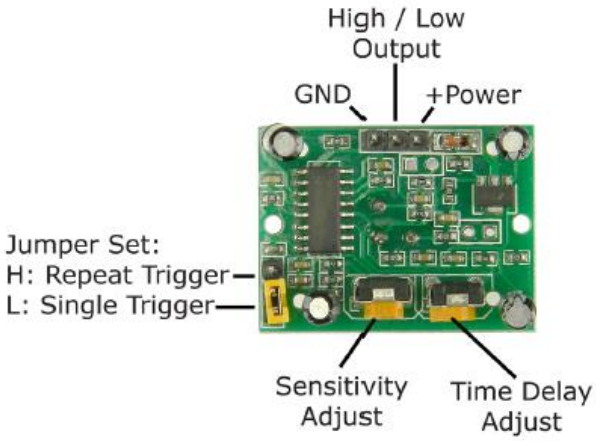
Os seguintes materiais são necessários para utilização do microcontrolador:

- Cartão SD de 32GB;
- Cabo micro *usb* com comum para utilizar com fonte de 5V;
- Cabo de rede;
- Roteador WIFI.

2.2.2 Sensores

- O sensor de presença PIR HC-SR501, Figura 2.7, foi escolhido pela alta confiabilidade e baixo custo, 15 reais, podendo ser ajustado para a sensibilidade escolhida, evitando assim detecção de presenças desnecessárias como animais.

Figura 2.7 – Pinagem do Sensor de parecença

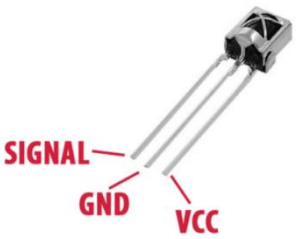


Fonte: ROBOCORE, 2018.

- Uma câmera IP é a melhor escolha devido sua alta resolução, alta compatibilidade com o sistema e fácil instalação, podendo ser sem fio. Uma *webcam* chinesa genérica 640x480 pixels foi escolhida por não ser necessário acessa-la pela internet, pelo baixo custo, 25 reais, e por fornecer uma imagem com qualidade mediana.
- O receptor infravermelho VS1838b, figura 2.8, funciona como sensor adquirindo os comandos de cada botão do controle remoto do condicionador de ar.

Ele possui um baixo custo, 2 reais, e as configurações genéricas necessárias para recepção de sinais (STAK, 2018).

Figura 2.8 – Receptor infravermelho VS1838b



Fonte: STAK, 2018.

2.2.3 Atuadores

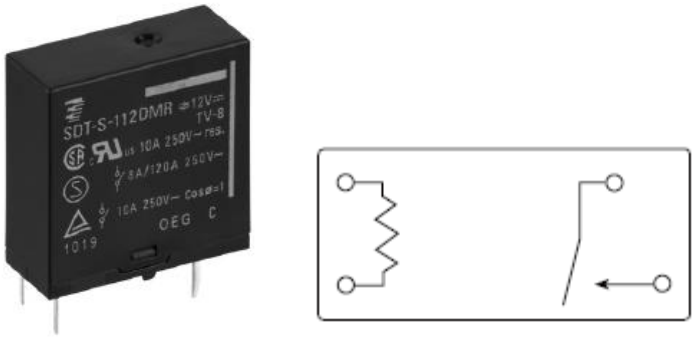
- A sirene foi utilizada como método de alarmar um possível intruso. Uma sirene de 12V é a melhor escolha por possuir alto volume de som e necessitar de poucos componentes, tais como um relé e algum tipo de chaveamento da bateria de 12V. No entanto uma de 3V foi escolhida por possuir menor preço, 5 reais, e atender à necessidade por se tratar de um protótipo.
- O LED infravermelho IR333C, figura 2.9, funcionará como atuador para controle do condicionador de ar. Ele foi escolhido para substituir um rele de desligamento para o circuito em questão, evitando assim danificar o equipamento e aumentando a flexibilidade do sistema. Por ser uma comunicação sem fio possui fácil instalação.

Figura 2.9 – LED infravermelho IR333C

Fonte: DIGIKEY, 2018.

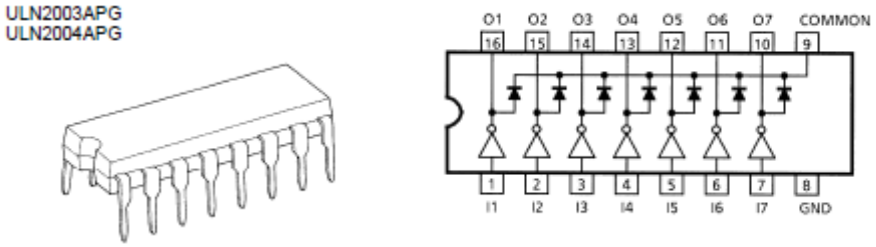
- O Relé SDT SS 109DM, figura 2.10 foi selecionado por trabalhar na faixa de tensão adequada para condicionadores de ar e requerer apenas chaveamento da bateria de 12V que ocorre por meio do Driver ULN2003APG, figura 2.11.

Figura 2.10 – Relé SDT SS 109DM



Fonte: TECONNECT, 2018.

Figura 2.11 – Driver ULN2003APG



Fonte: LCSC, 2018.

2.2.4 Bateria

Uma bateria de 12V foi utilizada para alimentar todos os equipamentos em caso de falta de energia. A bateria de nobreak de ATM POWER 5Ah foi escolhida por possuir um baixo custo, 38 reais. Um conversor de tensão 12V para 5V é necessário para alimentação do microcontrolador. Foi utilizado o circuito de um carregador *usb* automotivo comum para o tal.

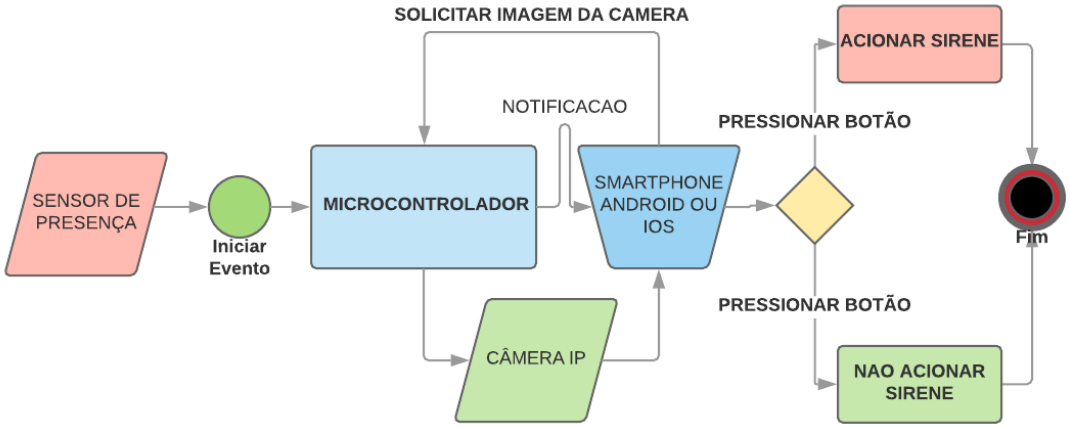
2.3 FLUXOGRAMA DAS FUNÇÕES

2.3.1 Fluxograma da função sistema de vigilância

Essa função consiste em realizar o envio de notificações pela internet para aplicativos *Android e IOS* caso um sensor de presença detecte movimento, contendo um *link* para uma *webcam* e botões que possibilitam acionar uma sirene. O sistema funciona integralmente mesmo durante falta de energia elétrica. Ele conta também

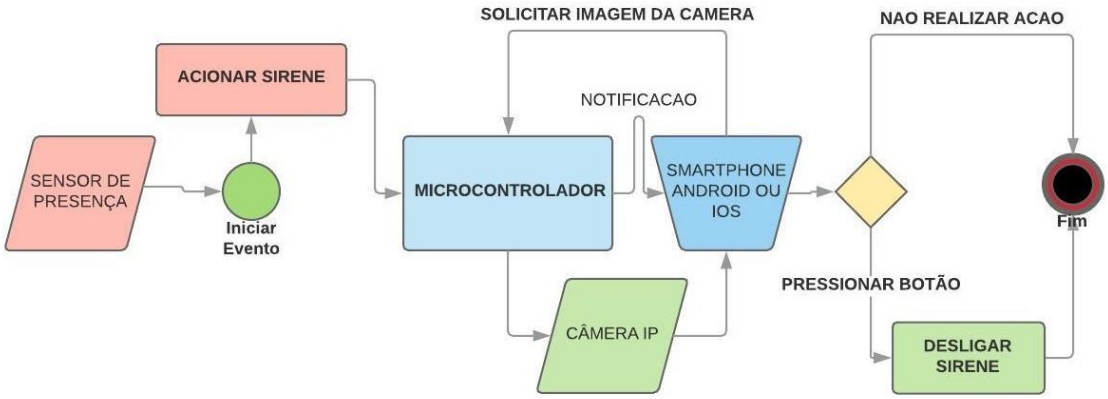
com o recebimento da posição de *GPS* do smartphone com intuito de lembrar ao usuário do acionamento da vigilância ou de acioná-lo automaticamente ao se distanciar da residência. Foram implementadas duas variações dessa função, elas se encontram nos fluxogramas das figuras 2.12 e 2.13.

Figura 2.12 – Fluxograma da função sistema de vigilância, alarme manual



Fonte: Próprio autor, 2018.

Figura 2.13 – Fluxograma da função sistema de vigilância, alarme automático



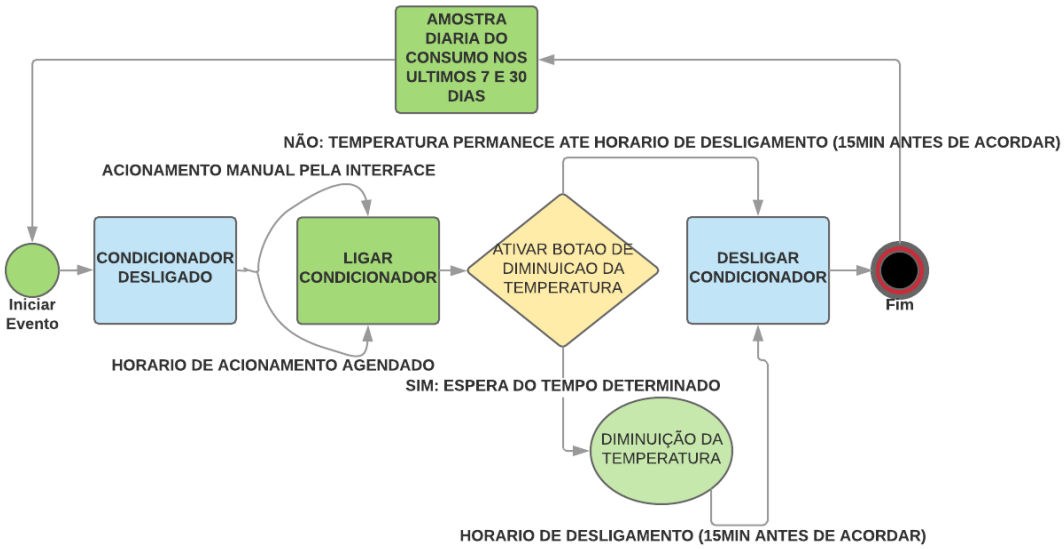
Fonte: Próprio autor, 2018.

2.3.2 Fluxograma da função automação de condicionadores

A função visa automatizar um condicionador de ar para que seja controlado por horário ou manualmente acessando o sistema, de modo que o sistema utilize os dados de funcionamento do condicionador e calcule diariamente o consumo dos

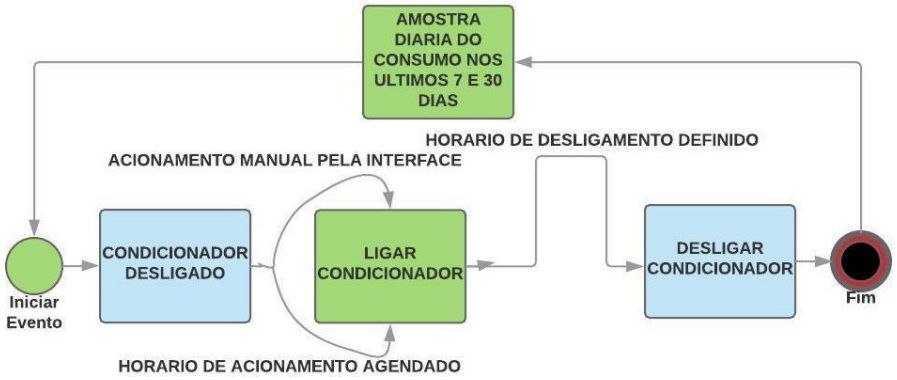
últimos sete dias e trinta dias. A função também permite realizar a diminuição da temperatura do condicionador após tempo decorrido do acionamento de uma chave no sistema. Um fluxograma da função se encontra na figura 2.14, no entanto o diagrama implementado no projeto se encontra na figura 2.15.

Figura 2.14 – Fluxograma da função automação de condicionadores



Fonte: Próprio autor, 2018.

Figura 2.15 – Fluxograma da função automação de condicionadores implementada



Fonte: Próprio autor, 2018.

3 IMPLEMENTAÇÃO DO PROJETO

A seguir estão descritos os subcapítulos da implementação do projeto:

1. CONFIGURACAO DO *SOFTWARE* – Configurações iniciais para facilitar a programação das funções;
2. CONFIGURAÇÃO DA FUNÇÃO SISTEMA DE VIGILÂNCIA - Instalação de componentes e programação e da função sistema de vigilância;
3. CONFIGURAÇÃO DA FUNÇÃO AUTOMAÇÃO DE CONDICIONADORES – Instalação de componentes e programação da função automação de condicionadores;
4. MONTAGEM DO CIRCUITO – Diagramas de ligação dos circuitos.

Para melhor acompanhamento do texto, os códigos de configuração do sistema que são inseridos no arquivo ‘configuration.yaml’ se encontram no apêndice A, os códigos de programação das funções que são inseridos no arquivo ‘automation.yaml’ se encontram no apêndice B, e códigos de personalização da interface que são inseridos no arquivo ‘groups.yaml’ se encontram no apêndice C.

3.1 CONFIGURAÇÃO DO *SOFTWARE*

O primeiro passo é baixar os *software*s necessários:

- Imagem do sistema compatível com o microcontrolador em questão, *Rasperry Pi* (HASSIO, 2018).
- *Software* “*Etcher*” para instalação da imagem no cartão SD (ETCHER, 2018).

Em seguida o cartão SD deve ser conectado ao computador e o programa ‘*Etcher’* iniciado, o mesmo não necessita instalação. No programa a imagem baixada deve ser localizada e o driver correspondente ao cartão SD selecionado, após o botão de início ser pressionado e a instalação finalizada o cartão deve ser retirado e conectado ao *Raspberry*.

O *Raspberry* deve ser conectado à fonte de alimentação e ao roteador por meio de cabo ethernet, após isso o microcontrolador começará a projetar um servidor web acessado pelo site ‘http://hassio.local:8123’, o mesmo deve ser

acessado em um *browser* cujo computador deve estar na rede LAN do mesmo roteador, WIFI ou Ethernet, a pagina deve exibir uma mensagem de atualização em andamento.

Após aproximadamente 40 minutos, dependendo da velocidade da internet, o sistema atualizado irá abrir a interface principal de configuração.

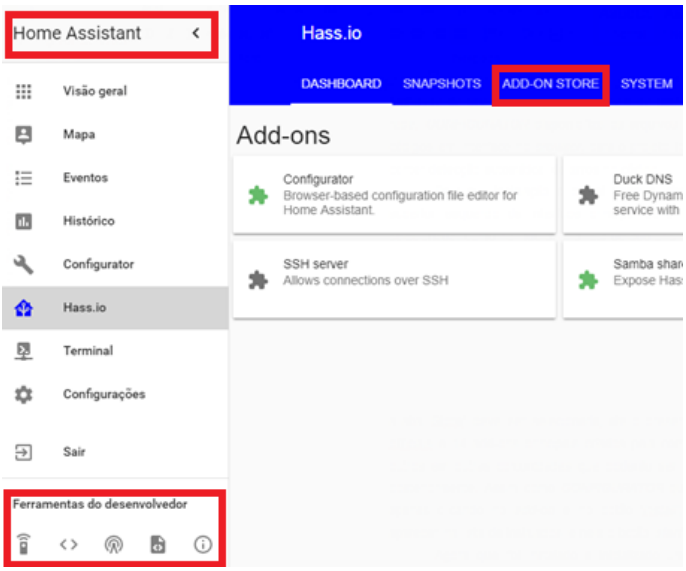
3.1.1 IDE (Ambiente de desenvolvimento integrado)

Para a configuração do sistema deverá ser feita a instalação do *Add-on* '*SAMBA SHARE'* ou '*CONFIGURATOR'* que possibilitam acessar os arquivos internos de configuração do sistema. *SAMBA SHARE* oferece facilidade de acesso ao registro por meio de pastas que podem ser acessadas pela rede. *CONFIGURATOR* disponibiliza os arquivos já em um ambiente de editor de códigos em interface no *browser*.

Para o projeto foi escolhido *CONFIGURATOR*, pois contém detecção automática de erros no código, lista com informações de todas as entidades, plataformas, serviços e eventos, link da biblioteca de componentes além da capacidade de botões de reiniciar seções do sistema.

Para sua instalação, o painel principal chamado '*Home Assistant'* no canto superior esquerdo da interface principal deve ser acessado e a aba '*Hass.io'* selecionada, como na figura 3.1, a lista de *Add-on*s instalados e as atualizações disponíveis é mostrada na aba 'DASHBOARD', a aba '*ADD* ON *STORE*' deve ser selecionada, até o presente momento existem 20 *Add-on*s oficiais e 14 *Add-on*s principais criados pela comunidade '*Github*', podendo existir outros em outras comunidades que poderão ser adicionados pelo método descrito posteriormente. Assim como *CONFIGURATOR* qualquer *Add-on* pode ser instalado apenas clicando no *Add-on* e no botão '*install'*. Após a instalação o *Add-on* ira aparecer na lista de instalados e nele o botão '*start*' deve ser pressionado.

Figura 3.1 – Local do painel principal

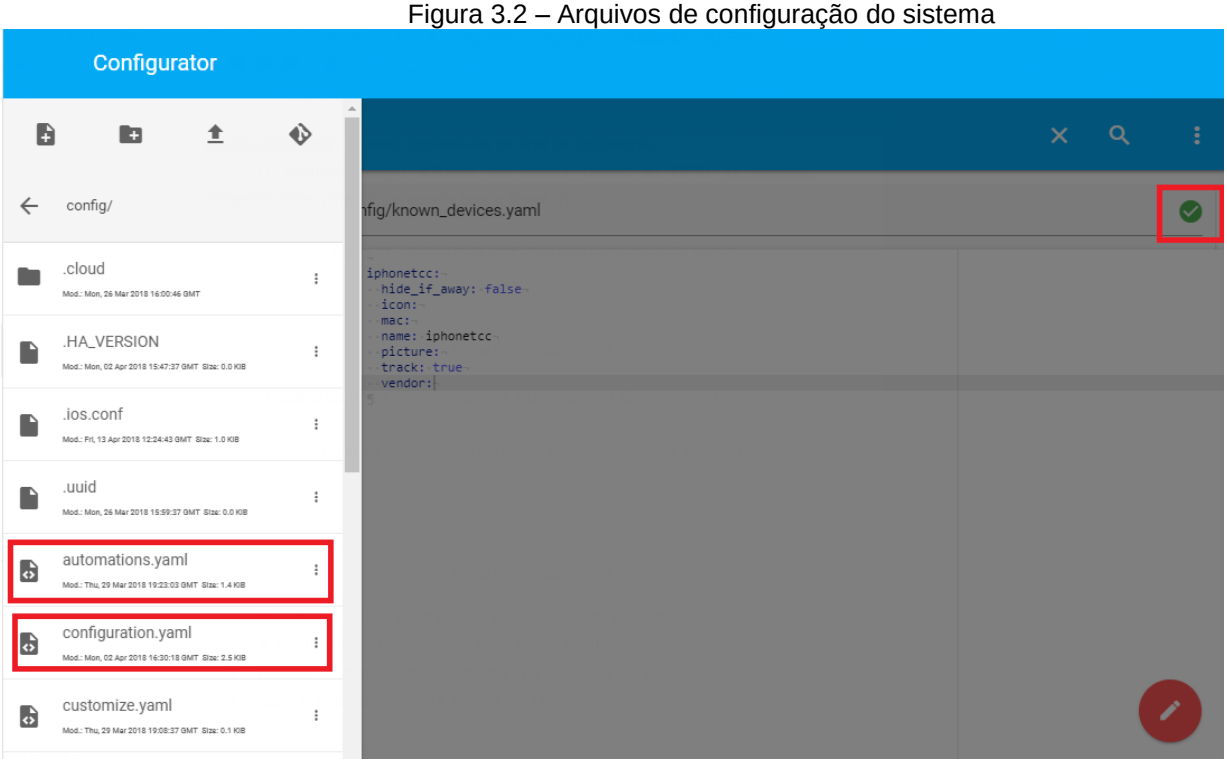


Fonte: Próprio autor, 2018.

Agora que foi instalado e inicializado um botão chamado '*Open* Web UI' aparecerá ao lado do botão '*start'* e ao aperta-lo o editor irá abrir em uma nova aba do navegador.

A figura 3.2 apresenta o local dos arquivos principais que serão editados no trabalho assim como o local do ícone de detecção de erros no código.

Figura 3.2 – Arquivos de configuração do sistema



Fonte: Próprio autor, 2018.

Para adicionar um atalho para *CONFIGURATOR,* o código da Figura 3.3 deverá ser copiado no final do documento chamado '*configuration.yaml*', que está localizado na Figura 3.2.

Figura 3.3 – Codificação em YAML para criação de ícone no menu

```
panel_iframe:
  configurator:
    title: Configurator
    icon: mdi:wrench
    url: http://hassio.local:3218
```

Fonte: Próprio autor, 2018.

Para que uma alteração ao arquivo 'configuration.yaml' tenha efeito, o sistema deve ser reiniciado abrindo o 'painel principal', selecionando 'Configuration', depois 'General' e depois o botão 'RESTART'. Agora o editor irá aparecer no 'painel principal' e irá abrir na mesma aba do navegador.

3.1.2 Servidor *Web*

Para acesso remoto de qualquer rede um servidor web deve ser criado. O 'Hass.io' proporciona uma instalação simples do servidor chamado '*Duckdns'*, que traduz o *IP* público do roteador em um *URL, ele tem suporte para IP* dinâmico e criptografia.

Para isso o roteador deve ser configurado, uma '*Port'* deve ser aberta para que o endereço *IP* do *Raspberry* possa ser acessado pela rede WAN ao invés de somente pela LAN. Nas configurações do roteador ao abrir a aba '*Port Forwarding',* ou Encaminhamento de Portas, uma janela com os mesmo campos que a Figura 16 deve ser mostrada, uma nova Porta deve ser adicionada preenchendo o campo '*IncomingPort'*, ou '*ExternalPort',* com o valor 8123 e o campo '*Trigger Port', ou 'InternalPort'* com 8123, no campo do *IP* digitar o *IP* do Rasperry, no campo protocolo selecionar 'AMBOS', de acordo com figura 3.4.

Figura 3.4 – Exemplo de abertura de uma "Port"

NO.	Porta de início- Porta de fim		LAN IP	Protocolo	Habilitar	Deletar
1.	8123	8123	192.168.0. 108	Ambos	✔	
2.			192.168.0.	TCP		

Fonte: Próprio autor, 2018.

Para achar o *IP* do *Raspberry* pode ser usada a aba lista de clientes no roteador ou instalar o aplicativo ‘*Ping*’ para *IOS* ou *android*.

Para confirmar que o procedimento foi bem sucedido o endereço ‘http://<IP público do roteador>:8123’ deverá ser acessado no *browser* fazendo com que a interface do *Home Assistant* seja mostrada.

O *IP* público pode ser encontrado facilmente em sites que mostrem seu ip (WHATIS, 2018).

É possível que o endereço anterior não funcione na mesma rede LAN onde o *Raspberry* está ligado, que é o caso de o roteador usado não suportar ‘*loopback’.* Nesse caso para o acesso na rede LAN o endereço ‘http://hassio.local:8123’ deve ser utilizado.

Caso o roteador seja usado em conjunto com um modem, como no caso de um modem da oi, a porta aberta anteriormente ainda não poderá ser acessada a menos que outro ‘Port Fowarding’ seja aberta nas configurações do modem, nesse caso a porta de entrada e saída é a mesma no entanto o endereço que deverá ser adicionado é o endereço WAN do roteador, esse processo deve ser feito pra todas as portas abertas.

Por último uma conta deve ser criada no site do ‘Duckdns’ (DUCKDNS, 2018). Após o ‘*login’* na conta criada um ‘*token’*, número serial, é fornecido e na mesma página no campo ‘*sub domain’* digitar um nome do domínio qualquer e pressionar ‘*Add domain’*, no domínio que irá aparecer abaixo digitar o *IP* público do roteador e pressionar ‘*update ip*’, com isso o domínio foi configurado.

Agora a interface poderá ser acessada em qualquer rede pelo endereço ‘http://< Nome do domínio>.duckdns.org:8123’.

Em seguida o servidor deve ser criptografado pra prevenir acesso de programas maliciosos, o ‘*Add-on*’ chamado ‘*Duckdns*’ pode criar certificados de criptografia que são renovados a cada três meses. O ‘*Add-on*’ deve ser instalado do mesmo modo que *CONFIGURATOR*, ao abri-lo as linhas da figura 3.5 irão aparecer.

Figura 3.5 – Janela de configuração do *Add-on* Duckdns

Fonte: Próprio autor, 2018.

Os seguintes campos deverão ser editados:

- accept_terms: ‘true’ para ativar a criptografia ou ‘false’ para desativar;
- token: número fornecido no site do servidor;
- domains: nome do site.

Mesmo sem criptografia o *Add-on* deve ser instalado com o valor ‘false’ no campo acima para que o domínio possa ter o endereço IP público dinâmico do roteador atualizado automaticamente no site.

Antes de criar os certificados uma na porta deve ser aberta, ‘*Port Forwarding’,* ou Encaminhamento de Portas deve ser acessada, uma nova Porta deve ser adicionada preenchendo o campo ‘*Trigger Port’*, ou ‘*External Port’,* com o serviço 80 e ‘*Internal Port’* com 80, no campo do *IP* digitar o *IP* do Rasperry.

Depois no *Add-on* clicar em ‘*save’* e então em ‘start’, esperar 4 minutos aproximadamente e no final da página clicar em ‘*refresh log’* para verificar se o certificado foi criado com sucesso, se na última linha do log contiver a palavra ‘*Done*’ os certificados foram criados.

Caso ocorra um erro é possível que o roteador não permita o uso da porta 80, então a porta deverá ser substituída pelo número 443, isso limitara as atualizações do certificado, mas permite a criação.

Após a criação dos certificados o código contido nas linhas 1 a 5 do apêndice A foi adicionado ao arquivo ‘configuration.yaml’ abaixo do campo ‘http:’, para fazer utilização dos mesmos.

Uma nova porta 3218>3218 deve ser criada e o campo ‘SSL’ do aplicativo *CONFIGURATOR* de ser modificado para ‘true’ para acesso do *CONFIGURATOR* remotamente, o mesmo vale para outros *Add*-ons que usem uma porta como o ‘TERMINAL’ que permite acesso à linha de comando pela porta 7681.

Agora o *Raspberry* poderá ser acessado e configurado no endereço protegido ‘https://< Nome do domínio>.duckdns.org:8123’, e apenas por ele.

Caso o roteador não suporte ‘*LoopBack*’ o único meio de acesso pela LAN local é pelo endereço ‘https://<IP local do *Raspberry*>:8123’.

Já que o endereço ‘http:hassio.local:3218’ não pode mais ser acessado o acesso ao ‘*CONFIGURATOR*’ deve ser atualizado do código na figura 15 para o código contido nas linhas 7 e 11 do apêndice A.

Os certificados duram apenas 90 dias, a partir de quando serão atualizados automaticamente desde que a porta 80 esteja aberta.

O código contido nas linhas 17 a 20 do apêndice A pode ser adicionado ao ‘configuration.yaml’ para adicionar um sensor que indica a quantidade de dias restantes para atualização dos certificados da criptografia que serão renovados automaticamente, caso não sejam renovados automaticamente o processo de criação deve ser refeito manualmente.

3.2 CONFIGURAÇÃO DA FUNÇÃO SISTEMA DE VIGILÂNCIA

Para adicionar a função de sistema de vigilância primeiramente os sensores e atuadores necessários devem ser adicionados ao sistema adicionando algumas linhas de código ao registro ‘configuration.yaml’, apenas copiando-os no final do documento.

O código contido nas linhas 33 a 36 do apêndice A adiciona um sensor binário ao GPIO 17, também conhecido como pino 11.

O código contido nas linhas 38 a 42 do apêndice A adiciona dois atuadores aos GPIO 27 e 23, também conhecidos como pinos 13 e 16.

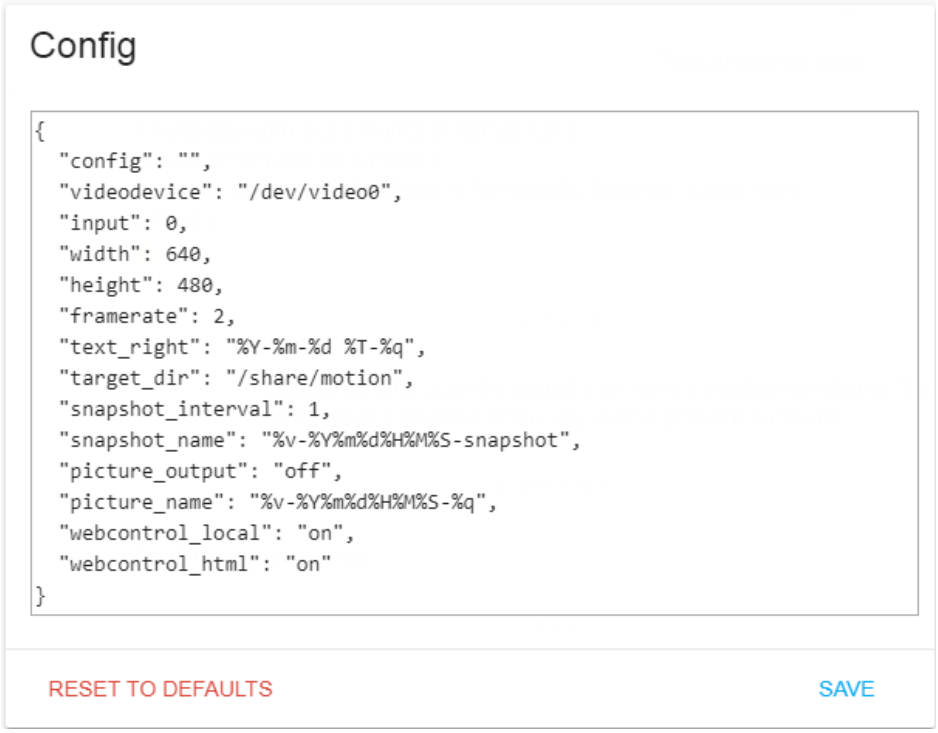
3.2.1 Câmera

A *webcam* possui uma resolução menor e é menos compatível com o sistema comparado a uma câmera IP, o *stream* de vídeo da *webcam* até o presente momento se limita a 1fps (frame por segundo).

Para instalação de uma *webcam* qualquer o *Add-on* '*motion*' deve ser instalado, o mesmo não consta na '*Add-on store*' mas pode ser encontrado na comunidade '*Github*' na página autor do *Add-on*, o *Add-on* pode ser instalado adicionando o endereço da página do *Github* (HERRHOFRAT, 2018) na caixa '*Add* new repositor by *URL*' localizada na '*Add-on store*' e pressionando '*ADD*' para adiciona-lo à loja.

Após selecionado e pressionado o botão '*install*' a configuração deve ser realizada como na figura 3.6.

Figura 3.6 – Janela de configuração do *Add-on Motion*



Fonte: Próprio autor, 2018.

Os seguintes campos deverão ser editados:

- Videodevice: porta *usb* que a câmera está conectada, podendo ser 0 ou 1;
- Snapshot_interval: número de segundos entre frames, 1 para 1fps;

- Picture_output: selecionar tipo de detecção de movimento ou desligar a detecção.

O *Add-on* já vem com função de detectar movimento por meio de variações na imagem da câmera e automaticamente armazenar como fotos quando ocorre detecção.

Para criação da entidade da câmera no sistema o código contido nas linhas 44 a 47 do apêndice A deve adicionado.

Uma câmera de celular *Android* foi utilizada como câmera *IP* no sistema, para sua adição código contido nas linhas 110 a 130 do apêndice A deve adicionado com o valor do *IP* do celular na rede, então o aplicativo '*IP Webcam*' deve ser instalado no *android*. A opção 'start server' ira ativar a câmera automaticamente no sistema.

3.2.2 IOS

Para a função de sistema de segurança é necessário o uso de notificações ao celular caso seja detectada uma presença inesperada, existem inúmeros componentes de notificação para smartphones, os componentes mais amigáveis, que enviam notificações por *sms* padrão ou chamada de voz, foram testados e nenhum dos mesmos ofereceu compatibilidade para o Brasil, sendo seis serviços de envio de *sms* e dois de chamada de voz, logo o único meio de envio no momento é pela internet.

Existem três meios de enviar notificações pela internet para as plataformas (*Android, IOS* e *Windows*) que são listados a seguir:

- *Email;*
- HTTP: por meio de *web push*;
- API *(AplicationPrograming Interface*): Comunicação entre aplicativo e servidor.

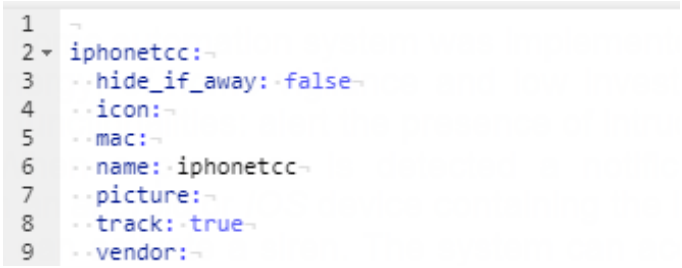
Foi verificado que o serviço que oferece maior eficiência seria envio de notificações por API.

O aplicativo *'Home Assistant Companion'* pode ser encontrado na *APPLESTORE* e é altamente integrado ao *Home Assistant* oferecendo diversas funcionalidades integradas como:

- Notificações avançadas com botões de ação e *stream* de vídeo;
- Acompanhamento da localização do *GPS*;
- Interface do sistema pode ser aberta no aplicativo.

Para utilização da plataforma IOS primeiramente o aplicativo 'Home Assistant Companion' deve ser instalado, e após conectar à mesma rede *LAN* que o *Raspberry* está conectado, o aplicativo reconhecera o endereço *IP* automaticamente e abrirá a interface interativa principal do *hassio*, nesse ponto o sistema irá reconhecer o aplicativo e instalar automaticamente todas as funções interativas que irão aparecer na interface, caso não apareçam o código da figura 3.7 deverá ser adicionado ao arquivo 'known-devices.yaml'.

Figura 3.7 – Codificação em YAML para adição das entidades do celular



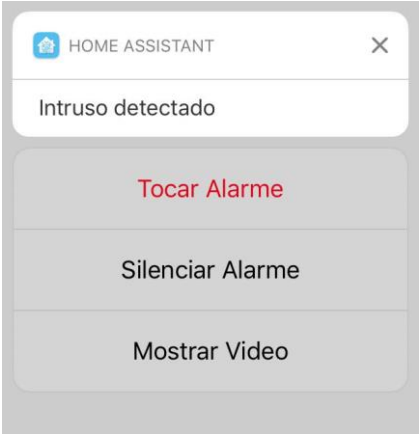
Fonte: Próprio autor, 2018.

Aonde a palavra na linha 02 deverá ter o mesmo nome definido no dispositivo.

Agora é preciso configurar o sistema com códigos chamados '*automations*' usados na criação de funções, seu código deve ser copiado para o final do arquivo chamado 'automation.yaml'.

Os códigos para adicionar notificações serão configurados a seguir. Em uma notificação é permitido de um a quatro botões de ação, cada botão ativa uma automação diferente, como mostra a figura 3.8.

Figura 3.8 – Exemplo de notificação mostrando botões de ação



Fonte: Próprio autor, 2018.

Primeiro uma categoria cujo código está contido nas linhas 49 a 62 do apêndice A deve ser adicionada no arquivo 'configuration.yaml' abaixo da linha "*ios:*", que pode ou não ter sido criada automaticamente.

Agora uma automação de notificação deve ser criada que use a categoria criada, cujo código está contido nas linhas 1 a 20 do apêndice B, o mesmo deve ser copiado no final do arquivo 'automation.yaml'.

Em seguida as automações pra cada botão devem ser criadas, o código que está contido nas linhas 22 a 38 do apêndice B enviará o vídeo da câmera como notificação. O código contido nas linhas 40 a 52 do apêndice B soará o alarme. O código contido nas linhas 55 a 67 do apêndice B desligará o alarme:

3.2.3 ANDROID

O sistema *Android* não possui aplicativo API oficial para acesso à interface do sistema nem para envio de notificações, dois aplicativos foram testados e serão apresentados a seguir.

Os aplicativos a seguir oferecem o maior número de funcionalidades e foram testados para a plataforma *Android*:

- '*Pushbullet*': realiza envio de notificações e arquivos por API para aplicativos *android, ios, Windows* e para extensões de navegadores *firefox, chrome, opera e safari*, além de permitir visualizar qualquer notificação ou ligação recebida no smartphone a partir dos outros dispositivo programado.
- '*Pushsafer*': realiza envio de notificações para aplicativos *Android, ios e*

Windows por API*, http* ou *email* e para extensões de navegadores *firefox, chrome*, além de configuração da notificação da mensagem.

O aplicativo '*Pushsafer*' oferece envio de mensagens por vários canais, o método de notificação escolhido foi por API.

O servidor de notificações '*Pushsafer*' oferece um aplicativo capaz de fornecer mensagens personalizadas, no entanto o serviço gratuito só oferece 25 mensagens por mês.

Uma conta de *e-mail* deve ser cadastrada no site do servidor (PUSHSAFER, 2018), após o *login* estará disponível uma chave privada que deverá ser copiada. O aplicativo 'PUSHSAFER' deverá ser instalado no celular *android* e efetuado o *login* criado anteriormente, a seguir o nome 'tccandroid' foi definido e o dispositivo foi reconhecido no site do servidor.

Agora é preciso instalar a componente no sistema adicionando o código contido nas linhas 64 a 67 do apêndice A, dentro do campo 'notify' no arquivo 'configuration.yaml', os valores da linha 67 são de uma chave serial fictícia que deve ser substituído pela chave entregue no site.

O código para adicionar uma notificação está contido nas linhas 69 a 91 do apêndice B.

O aplicativo 'Pushbullet' oferece envio de mensagens por API, no entanto o serviço gratuito permite no máximo 100 mensagens e 25MB para envio de arquivos por mês.

Uma conta de *email* deve ser cadastrada no site do servidor (PUSHBULLET, 2018), após o *login* estará disponível na aba 'settings/Account' uma chave privada que deverá ser copiada. O aplicativo 'PUSHBULLET' deverá ser instalado no celular *android* e efetuado o *login* criado anteriormente, após cada *login* por um dispositivo diferente um novo dispositivo será reconhecido no site do servidor, mais de uma conta de *email* pode ser vinculada.

Agora é preciso instalar a componente no sistema adicionando o código contido nas linhas 68 a 70 do apêndice A, dentro do campo '*notify*' do arquivo 'configuration.yaml', os valores da linha 77 são de uma chave serial fictícia que deve ser substituído pela chave entregue no site.

O código para adicionar uma notificação está contido nas linhas 93 a 107 do apêndice B.

Para adicionar funções que utilizem posição de GPS do *smartphone* o aplicativo ‘GPSLogger’ deve ser instalado para o *android* e o aplicativo ‘*Home assistant companion*’ para *IOS.*

No aplicativo IOS é necessário apenas que a opção posição das configurações esteja ativada, no aplicativo *android* na configuração chamada ‘Detalhes de Log’ deve ser desativadas todas as opções de log e ativada a opção ‘*Log* para *url*’, fazendo adição da *url* ‘https://[endereço do sistema]:[Porta]/api/gpslogger?latitude=%LAT&longitude=%LON&device=[Nome]&accuracy=%ACC&battery=%BATT&speed=%SPD&direction=%DIR&altitude=%ALT&provider=%PROV&activity=%ACT’ deve ser adicionada.

Para instalar a componente de recebimento da localização as linhas 72 e 73 do apêndice A devem ser adicionadas.

Para adicionar um local, zona, para usar como referência o código contido nas linhas 75 a 80 do apêndice A deve ser adicionado.

O código contido nas linhas 109 a 131 do apêndice B cria uma automação para notificar o usuário quando ocorre afastamento de 250 metros do local da residência lembrando a ativação do alarme de vigilância e desligamento do condicionador.

O código contido nas linhas 133 a 146 do apêndice B cria uma automação para desligar o condicionador automaticamente quando ocorre afastamento do local.

O código contido nas linhas 148 a 167 do apêndice B adiciona uma automação para acionamento automática da sirene quando ocorre detecção de presença.

O código contido nas linhas 22 a 27 do apêndice C realiza a união das automações listadas para fazer uma automação com função de alarme de vigilância, aonde a câmera se encontra na interface de usuário ou enviada por mensagem para aplicativo IOS, não é o caso pois webcam não são suportadas pelo aplicativo. Essa união também pode ser realizada com a adição de todos os códigos na mesma automação, no entanto por fim de simplicidade e facilidade de personalização da função a união em grupo de automações foi escolhida.

O código contido nas linhas 29 a 33 do apêndice C realiza a mesma união listada acima com exceção da automação de sirene automática.

O código contido nas linhas 1 a 8 do apêndice C adiciona os dois grupos de automações anteriores e outras entidades com relação à vigilância.

3.3 CONFIGURAÇÃO DA FUNÇÃO AUTOMAÇÃO DE CONDICIONADORES

Para fazer o controle do condicionador de ar a melhor opção, que permite controle da temperatura, é por meio da tecnologia sem fio infravermelho, o sistema, até o momento, depende da componente LIRC (*Linux Infrared Remote Control).* LIRC é uma componente que permite decodificar sinais infravermelhos de muitos controles remotos e controlar ações dependendo do botão pressionado por meio de um receptor IR e transmissor IR. No entanto essa componente, que já possui documentação, até o momento tem sua instalação disponível apenas nas versões de instalação '*Python Virtual Enviroment'* e '*Hassbian*' do sistema, pelo método de instalação '*Hass.io*' não há possibilidade pois os comandos em *cmd* necessários para a instalação não são suportados, portanto o controle do condicionador de ar foi realizado por meio de chaveamento por relé que é controlado pela porta GPIO 23 criada pelos códigos presentes no item 3.2.

Para fazer o controle por horário do condicionador o código contido entre as linhas 92 a 98 do apêndice A adiciona duas entradas de horário.

Um sensor de horário e de data deve ser instalado com o código contido entre as linhas 21 a 24 do apêndice A, os mesmos irão aparecer na interface de usuário.

Uma automação para controle do condicionador por horário se encontra no código contido entre as linhas 169 a 189 do apêndice B.

Para realizar cálculo de consumo, o sistema precisa gravar o número de horas que o condicionador esteve ligado, código contido na linha 100 do apêndice A grava o estado de todas as entidades, após isso o sensor contido entre as linhas 25 a 30 do apêndice A realiza a soma das horas no estado ligado nos últimos 30 dias.

O código contido entre as linhas 102 a 108 do apêndice A cria um gráfico do estado do condicionador por hora nos últimos 30 dias.

Para o cálculo do consumo é necessário a potencia media mensal a cada hora que é fornecida no equipamento de condicionador, a entrada do valor é criada pelo código contido entre as linhas 82 a 90 do apêndice A.

Uma automação para apresentar uma mensagem na interface com o consumo mensal apresentado diariamente se encontra no código contido entre as linhas 191 a 206 do apêndice B.

Uma automação para apresentar uma mensagem na interface com o

consumo mensal todo dia 1 se encontra no código contido entre as linhas 208 a 227 do apêndice B.

O código contido entre as linhas 11 a 20 do apêndice C organiza os itens da interface de usuário relacionados a essa função.

Os códigos dos apêndices podem ser copiados inteiramente nos seus arquivos respectivos devendo se atentar aos seguintes itens:

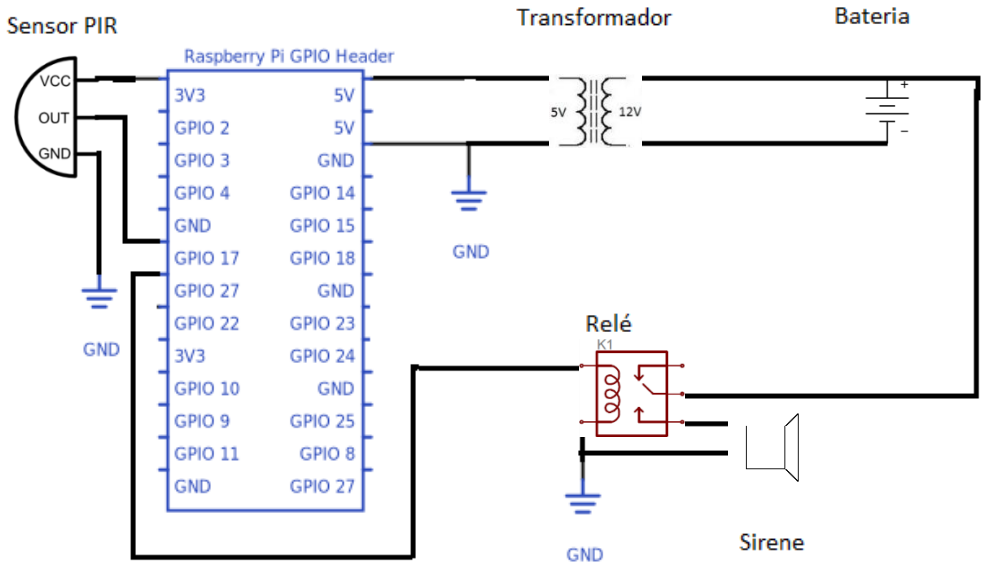
- Notações próximas do inicio ao fim de *templates* devem ser removidas;
- *Templates* e textos de mensagens em mais de uma linha, que se encontram na linha seguinte do símbolo '>', devem iniciar após dez espaços do inicio da linha que se encontram;
- Valores de domínio de internet devem ser adicionados, no lugar marcado, nos itens 'url';
- O campo 'http:' deve ser copiado apenas após a criptografia ser adicionada e os certificados criados com sucesso, lembrando de mudar o campo 'SSL' para 'yes' em *add-ons* como CONFIGURATOR;
- Após cada mudança no arquivo yaml o botão de 'checar configuração' localizado em configurações deve ser pressionado.

3.4 MONTAGEM DO CIRCUITO

3.4.1 circuito da função de sistema de vigilância

A figura 3.9 ilustra o esquema de ligação com o uso da sirene de 12V, com os pinos definidos, a *webcam* é ligada diretamente na entrada *usb* do *Raspberry* e a câmera *android* é interligada pela rede *WIFI*.

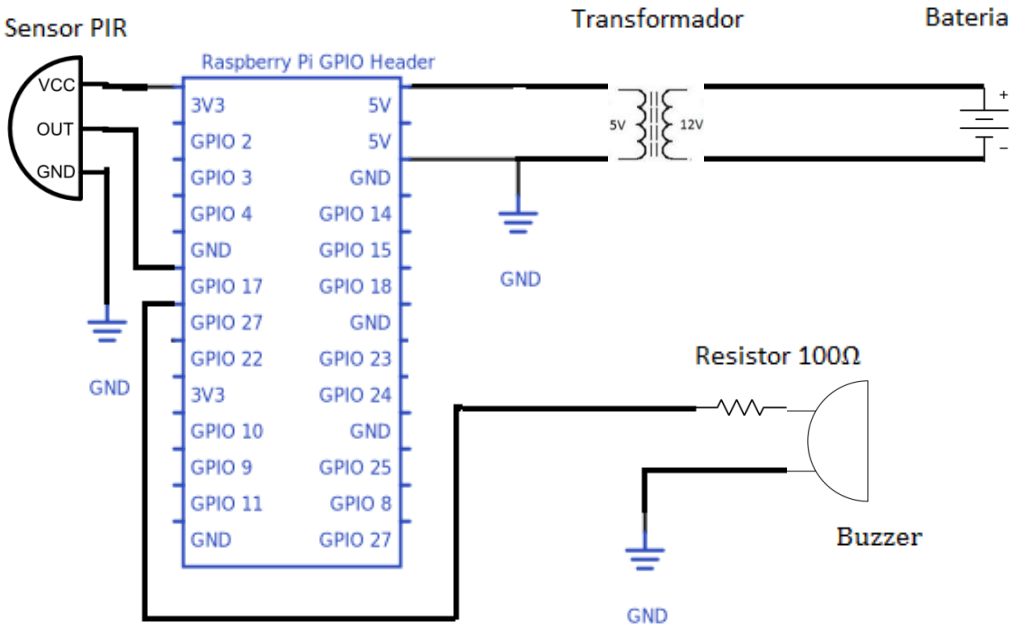
Figura 3.9 – Esquema de ligação ilustrativo Ideal



Fonte: Próprio autor, 2018.

A sirene de 12V foi substituída por uma de 3,3V como definido no item 2.2.3 logo a Figura 3.10 ilustra o circuito implementado.

Figura 3.10 – Esquema de ligação ilustrativo definido

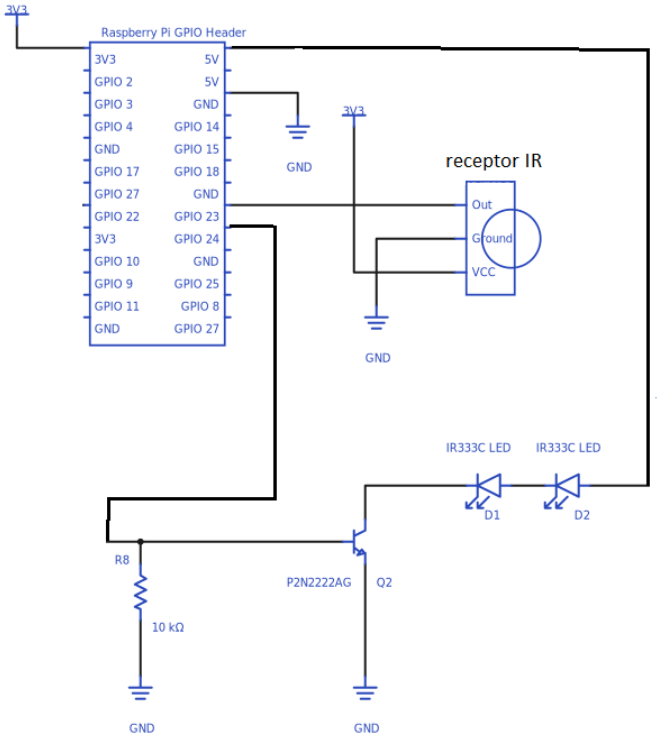


Fonte: Próprio autor, 2018.

3.4.2 circuito da função controle de condicionadores

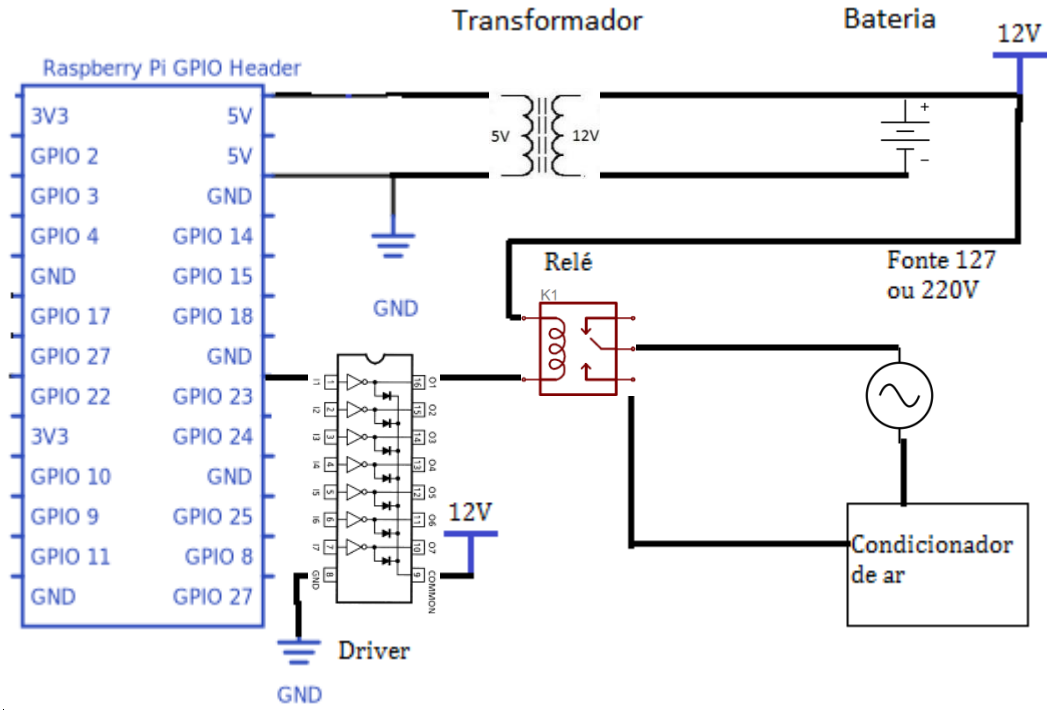
A Figura 3.11 ilustra o esquema de ligação com os pinos definidos para a função com controle por infravermelho. A Figura 3.12 ilustra o esquema de ligação com os pinos definidos para a função com controle por relé. O relé utilizado suporta apenas condicionadores de 15000Btu's do tipo janela ou 18000Btu's do tipo split, podendo ser substituído por outro mais potente.

Figura 3.11 – Esquema de ligação ilustrativo



Fonte: ALEXBA, 2018.

Figura 3.12 – Esquema de ligação ilustrativo



Fonte: Próprio autor, 2018.

4 RESULTADOS OBTIDOS

Os resultados foram obtidos a partir dos seguintes testes:

- Teste do *software* da função sistema de vigilância;
- Teste do *software* da função controle de condicionadores;
- Teste do *hardware* da função sistema de vigilância;
- Teste do *hardware* da função controle de condicionadores.

4.1 TESTE DO *SOFTWARE* DA FUNÇÃO SISTEMA DE VIGILÂNCIA;

Para realização da validação do *software* implementado foi utilizada uma entrada virtual, uma chave no lugar do sensor, que é mostrada na figura 4.1, para descartar possíveis erros nos sensores. A chave simulava o sensor de presença, seu acionamento foi feito repetidamente 15 vezes.

Figura 4.1 – Chave na interface do sistema

Fonte: Próprio autor, 2018.

Sempre que a chave era acionada uma notificação era enviada ao celular com sistema *IOS* que estava funcionando em rede 3G e com o aplicativo instalado rodando no modo minimizado, mesmo com o celular em *standby* todas as notificações chegaram sem erros, com um atraso de 2 a 3 segundos, com o som escolhido e com os botões de ação trabalhando normalmente para o chaveamento da chave virtual da sirene, segundo figura 4.2 aonde um *GIF* se encontra no lugar do *stream* da câmera.

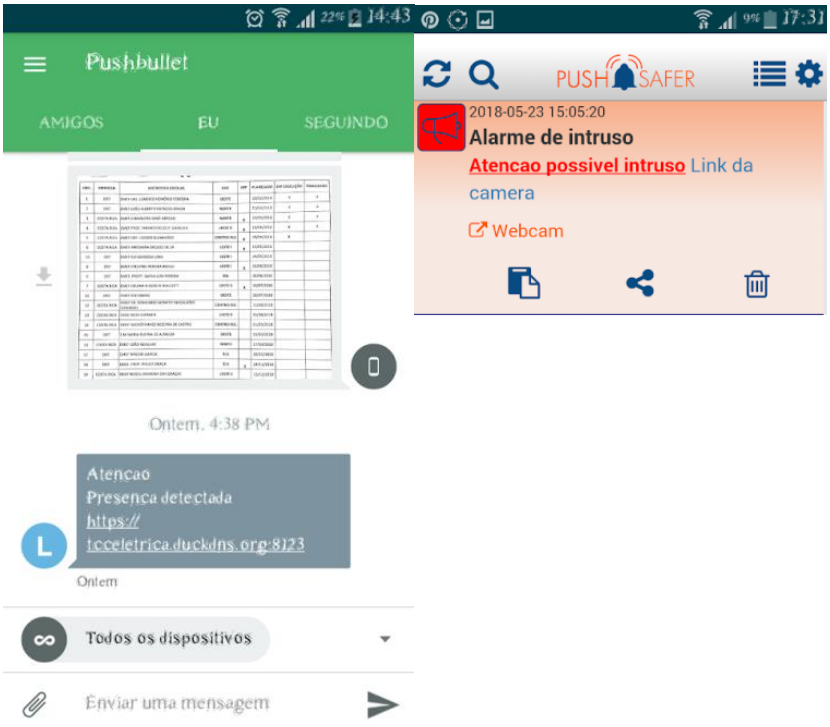
Figura 4.2 – Notificação de ativação do sensor

Fonte: Próprio autor, 2018.

Ao apertar o botão ‘Tocar Alarme’ a variável do atuador Sirene foi acionada como esperado e em seguida uma notificação chegou afirmando a mudança e apresentando os botões de ação. Ao pressionar ‘Silenciar Alarme’ a variável é desativada.

O mesmo teste foi realizado para ambos os aplicativos *android* e resultado obtido teve atraso semelhante, porem sem a presença dos botões de ação como mostra a figura 4.3. O aplicativo Pushbullet tem um som relativamente baixo que não pode ser mudado.

Figura 4.3 – Notificação de ativação do sensor



Fonte: Próprio autor, 2018.

A câmera que se encontra na interface do usuário, mostrada na Figura 4.4, apesar de fornecer uma imagem de baixa resolução permite uma visualização adequada a 1fps, *frames* por segundo. A câmera *android* permite a escolha da resolução, que deve ser escolhida de acordo com a velocidade da internet. Quando a adequada é escolhida o vídeo apresenta alto fps, por volta de 10.

Figura 4.4 – Local da função na interface de usuário

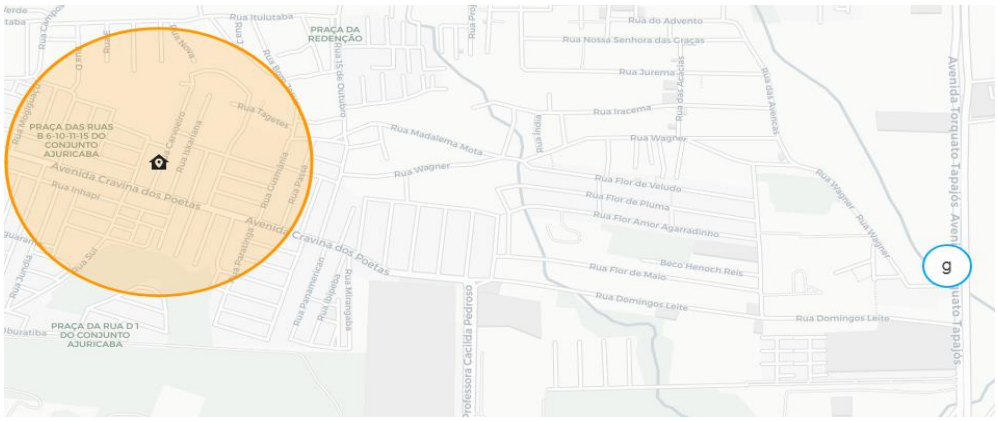
Fonte: Próprio autor, 2018.

O aplicativo de rastreamento de *gps* do *android* permite configuração de precisão da localização e de intervalo de envio da localização fazendo com que o aplicativo funcione em segundo plano sem consumir muita bateria e mantendo a precisão desejada.

O aplicativo para *IOS* atendeu a demanda de precisão, no entanto o intervalo de envio da localização é limitado intencionalmente pela *Apple* para um envio a cada 500m de deslocamento, garantindo ótima economia de bateria mas necessitando mudar o raio da zona da residência para 600m.

O aplicativo de rastreamento, reconhecido na interface de usuário como na Figura 4.5, foi testado usando um aplicativo de *GPS* falso, foram realizadas 10 rotas entrando e saindo do raio da residência, todas sem erros de medição.

Figura 4.5 – Mapa de rastreamento na interface de usuário



Fonte: Próprio autor, 2018.

O sistema se mostrou seguro, por utilização de conexão criptografada, eficiente, com baixo atraso na realização das automações, e confiável, por funcionar sem erros mesmo funcionando com acesso à distância e com o *smartphone* em *standby*.

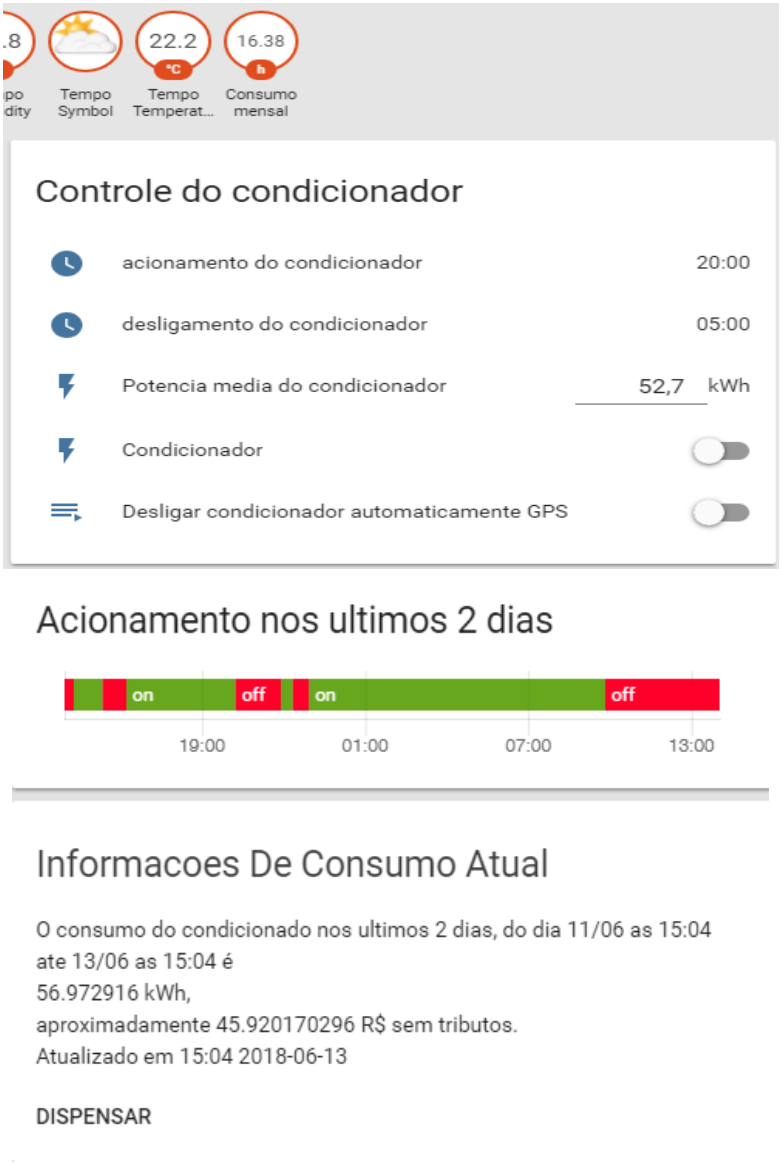
4.2 TESTE DO *SOFTWARE* DA FUNÇÃO CONTROLE DE CONDICIONADORES

Para realização da validação do *software* implementado foram preenchidas as entradas de horário, para acionamento e desligamento, e a entrada de consumo médio mensal, cujo valor é de um condicionador split de 24000Btu’s, como na Figura 4.6.

Os dados de horário do condicionador nos últimos 2 dias foram coletados de acordo com o estado da chave de controle do condicionador. A chave virtual de controle do acionamento do condicionador foi observada e ocorreu o acionamento e desligamento nos horários definidos.

A notificação de consumo era atualizada a cada hora quando o sensor de consumo fazia nova contagem do total de horas nos últimos 2 dias, horas em que a chave estava desligada não eram somadas, o valor de 16,38h foi somado. Então esse valor em conjunto do valor de consumo médio mensal foi utilizado no calculo do consumo real. Um gráfico do estado da chave foi plotado nos 2 dias testados.

Figura 4.6 – Local da função na interface de usuário

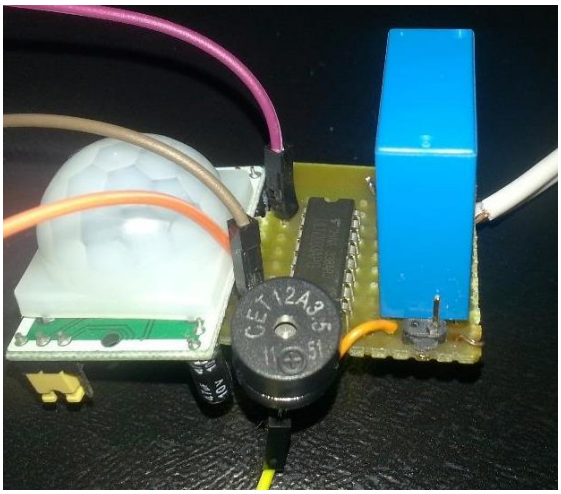


Fonte: Próprio autor, 2018.

4.3 TESTE DO *HARDWARE* DA FUNÇÃO SISTEMA DE VIGILÂNCIA

O circuito da Figura 3.10 foi montado e se encontra nas figuras 4.7 e 4.8. O teste foi realizado fazendo movimentos para acionar o sensor, ao fazê-lo com o sistema de vigilância ativo o *buzzer* acionou e as notificações foram enviadas aos *smartphones* de acordo com o item 4.1.

Figura 4.7 – Circuito de sensor de presença e *buzzer*



Fonte: Próprio autor, 2018.

Figura 4.8 – Circuito com bateria e transformador

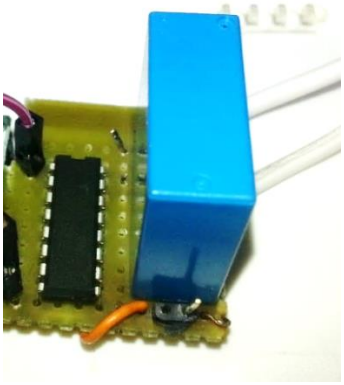
Fonte: Próprio autor, 2018.

O uso da bateria possibilitou o funcionamento de todas as funções descritas durante quedas de energia.

4.4 TESTE DO *HARDWARE* DA FUNÇÃO CONTROLE DE CONDICIONADORES

O circuito do rele, segundo figura 3.12, foi montado e se encontra na figura 4.9.

Figura 4.9 – circuito de controle por relé

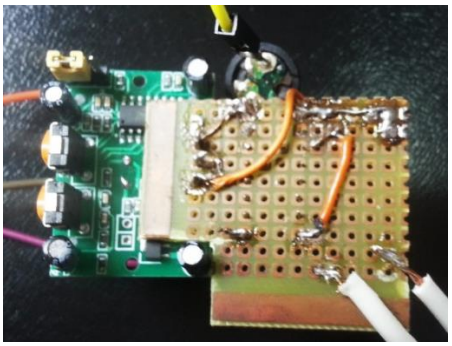


Fonte: Próprio autor, 2018.

O teste no circuito foi feito por horário e manualmente e apresentou um atraso de 1 segundo no acionamento do relé, o acionamento do relé necessita da bateria de 12V.

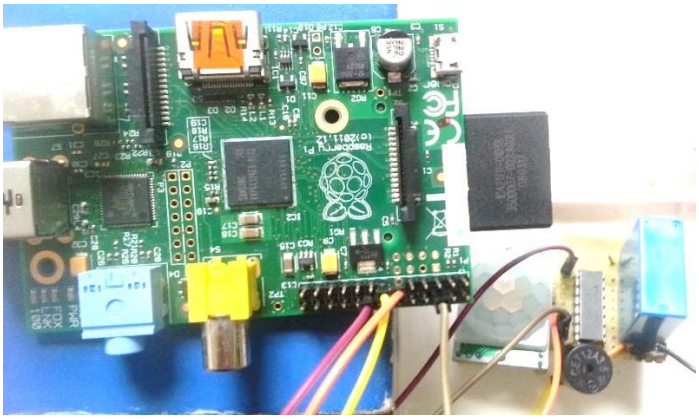
A Figura 4.10 apresenta a placa conjunta dos circuitos, e a Figura 4.11 apresenta a interligação de todos os circuitos na placa.

Figura 4.10 – placa de todos os circuitos



Fonte: Próprio autor, 2018.

Figura 4.11 – interligação dos circuitos



Fonte: Próprio autor, 2018.

4.5 ANÁLISE E COMPARAÇÃO DE CUSTOS

O custo total do projeto foi analisado por meio da tabela 01. A placa microcontroladora foi substituída pela versão superior pelos motivos citados no item 2.2.1, a sirene 3V foi substituída por uma de 12V citada no item 2.2.3 no entanto foi utilizado o preço da serene 12V.

Tabela 4.1 – Preço dos materiais utilizados

MATERIAL	PRECO	
BATERIA ATM POWER 12V/5Ah	R$	38,00
CARREGADOR USB VEICULAR	R$	10,00
SIRENE 12V 20W INTELBRAS ECP	R$	28,00
LED INFRAVERMELHO IR333C	R$	2,00
RECEPTOR IR VS1838b	R$	3,00
SENSOR PIR	R$	15,00
*WEBCAM*ERA	R$	25,00
RASPBERRY PI 2 MODEL B	R$	170,00
RELÉ 12VDC SDT SS 109DM	R$	7,00
DRIVER ULN2003APG	R$	3,00
FONTE 127/12V	R$	60,00
FONTE 127/5V	R$	5,00
CARTÃO SD DE 32GB classe 10	R$	38,00
PREÇO TOTAL	R$	404,00
PREÇO TOTAL COM 10% DE CUSTOS DE IMPLEMENTAÇÃO	R$	444,40

Fonte: Próprio autor, 2018.

O custo total do projeto com a adição de 6 novos circuitos e de uma câmera *IP*, mais custos de instalação se encontra na Tabela 4.2

Tabela 4.2 – Preço dos materiais utilizados com adição de circuitos

MATERIAL	PRECO	
BATERIA ATM POWER 12V/5Ah	R$	38,00
CARREGADOR USB VEICULAR	R$	10,00
SIRENE 12V 20W INTELBRAS ECP	R$	28,00
LED INFRAVERMELHO IR333C	R$	2,00
RECEPTOR IR VS1838b	R$	3,00
SENSOR PIR	R$	15,00
WEBCAM	R$	25,00
CÂMERA IP	R$	130,00
RASPBERRY PI 2 MODEL B	R$	170,00
7 x RELÉ 12VDC SDT SS 109DM	R$	49,00
DRIVER ULN2003APG	R$	3,00
FONTE 127/12V	R$	60,00
FONTE 127/5V	R$	5,00
CARTÃO SD DE 32GB classe 10	R$	38,00
PREÇO TOTAL	R$	576,00
PREÇO TOTAL COM 10% DE CUSTOS DE IMPLEMENTAÇÃO	R$	633,00

Fonte: Próprio autor, 2018.

O custo total do projeto foi consideravelmente inferior ao preço de um sistema de automação residencial profissional com custo 9 vezes menor que o kit básico da empresa Schutz Automação citada, que oferece apenas automação de 3 circuitos e *home theater*.

O sistema oferece funções avançadas assim como muitos sistemas profissionais, possibilita vigilância a distancia com uso de webcam ou câmeras *ip* por um preço inferior a um sistema de vigilância de DVR. Os sistemas profissionais, no entanto, possuem atendimento 24h, funções com maior detalhamento e de fácil personalização.

CONCLUSÃO

Na realização deste trabalho foi aplicado o *software open source Home Assistant,* com intuito de reduzir o custo total do sistema, instalado em um microcontrolador para operação automatizada de protótipos de circuitos eletrônicos.

Foi apresentado um breve capitulo no método proposto para demonstração da linguagem utilizada pelo *software*, seguido de fluxogramas para ambas as funções pretendidas.

Em implementação do projeto foram expostos os códigos de configuração do sistema para realização do controle a distância e das funções de automação propostas, seguido dos esquemas de ligação dos dois circuitos eletrônicos.

Depois de finalizados foram realizados testes de *software*, nos aplicativos *Pushsafer, Pushbullet, Gpslogger, Home Assistant Companion* e *Home Assistant*, e de *hardware*, nos circuitos eletrônicos contendo sensores e atuadores. Os testes realizados no sistema apresentaram funcionamento adequado das funções pretendidas, sendo evidenciada a confiabilidade, segurança e eficiência do *software* aplicado, assim como o resultado do *hardware*.

Com base nos valores apresentados na tabela 01 e o exposto no item 1.1 do referencial teórico pode-se confirmar a hipótese de que é possível implementar um sistema funcional com custo inferior ao de mercado. O sistema implementado pode oferecer serviços de alta complexidade assim como outros que possuem alto custo no mercado, mantendo sua segurança e sem prejuízo da funcionalidade.

O uso da comunicação sem fio infravermelho, que era o método pretendido de controle do condicionador, no entanto não foi possível devido inexistência de *software* atual para implementação no sistema. Sugere-se, portanto, implementação dessa tecnologia após a criação novos softwares compatíveis.

Outras comunicações sem fio como o receptor RF KR2201-A ou o Relé *WIFI*

Esp8266, que permitem chaveamento simplificado de circuitos, podem ser implementadas aproveitando a tecnologia *WIFI* embutida em varias das placas microcontroladoras disponíveis atualmente.

Sensores de fumaça e de detecção de IP na rede, que já são suportados pelo sistema, podendo ser implementados futuramente para aumento da segurança e do sensoriamento, respectivamente.

O sistema oferece conexão com inúmeras tecnologias de leitura texto e detecção de voz, incluindo o *Google Assistant*, que pode facilitar ainda mais o controle do sistema.

REFERÊNCIAS BIBLIOGRÁFICAS

ALEXA. **Analise de websites.** Disponível em: <http://www.check-domains.com/website-analysis/website-analyzer.php>. Acesso em: 10 Maio. 2018.

ALEXBA. **Tecnical Blog**. Disponível em: < http://alexba.in/blog/2013/01/06/setting-up-lirc-on-the-raspberrypi/>. Acesso em: 10 Fev. 2018.

ALMEIDA, Rodrigo Carpim de; **Sistema de segurança interativo e a prova de quedas na rede elétrica.** Trabalho de conclusão de curso – Curso de Graduação em Engenharia Elétrica USP, São Carlos, 2016.

AURESIDE, **Associação Brasileira de Automação Residencial**. Disponível em: <http://www.aureside.org.br/>. Acesso em: 10 Abril. 2017.

AUTO. **Catálogo Automação Residencial Brasil.** Disponível em: < https://automacaoresidencialbrasil.com.br/>. Acesso em: 10 Maio. 2018.

BEGHINI, Lucas Bragazza; **Automação residencial de baixo custo por meio de dispositivos móveis com sistema operacional Android**. Trabalho de conclusão de curso – Curso de Graduação em Engenharia Elétrica – USP, São Carlos, 2013.

BORGES, L. P.; DORES, R. C., **Automação predial sem fio utilizando bacnet/zigbee com foco em economia de energia**. 2010, 76f. Trabalho de conclusão de curso - Curso de Graduação em Engenharia de Controle e Automação – UNB, Brasília, 2010.

BOXTECH. **Catalogo**. Disponível em: < https://shop.boxtec.ch/raspberry-model-rev-p-41604.html>. Acesso em: 20 Set. 2017.

BRUGNARI, A.; MAESTRELLI, L. H. M., **AUTOMAÇÃO RESIDENCIAL via WEB**. 2010, 36f. Trabalho de conclusão de curso - Curso de Graduação em Engenharia de Computação - PUC-PR, Curitiba, 2010.

BUBNIAK. **Gazeta do povo.** Disponível em: < http://www.gazetadopovo.com.br/imoveis/automacao-residencial-esta-mais-cessivel-351rpjeedgi5win9vowjei826>. Acesso em: 20 Set. 2017.

CARAUDIORJ. **Ricardo Freitas**. Disponível em: < https://www.caraudiorj.com/single-post/2017/01/03/Automa%C3%A7%C3%A3o-Residencial---Solu%C3%A7%C3%A3o-ou-Problema/>. Acesso em: 20 Set. 2017.

CEDOM. **Associação Espanhola**. Disponível em: < http://www.cedom.es/sobre-domotica/que-es-domotica>. Acesso em: 20 Fev. 2018.

CHRISATECH. **Chris Tech Blog**. Disponível em: < https://chrisatech.wordpress.com/2015/10/16/raspberry-pi-shutdown-switch/ >. Acesso em: 20 Set. 2017.

CONCEIÇÃO JUNIOR. **Redes sem Fio: Protocolo *Bluetooth* Aplicado em Interconexão entre Dispositivos**. Disponível em: <http://www.teleco.com.br/pdfs/tutorialredespbaid.pdf>. Acesso em: 10 Abril. 2017.

DIGIKEY. **Digikey.com**. Disponível em: < https://www.digikey.com/product-detail/en/everlight-electronics-co-ltd/IR333C/1080-1081-ND/2675572>. Acesso em: 10 Jan. 2018.

DUCKDNS. **Free dynamics DNS**. Disponível em: <https://www.duckdns.org>. Acesso em: 10 Jan. 2018.

EDITOR. **Editor de automações**. Disponível em: < https://www.home-assistant.io/docs/automation/editor/>. Acesso em: 20 Fev. 2018.

ELEMENT14. **Catalogo Element14**. Disponível em: < https://www.element14.com/community/community/raspberry-pi/raspberrypi2>. Acesso em: 20 Set. 2017.

ELETROBRAS. **Resolução ANEEL 2337**. Disponível em: < http://www.eletrobrasamazonas.com/cms/wp-content/uploads/2017/02/RESUMO-DAS-TARIFAS-ATUALIZADAS-RESOLU%C3%87%C3%83O-N%C2%BA-2337-31.10.17.pdf/>. Acesso em: 10 Mar. 2018.

ETCHER. **Etcher**. Disponível em: <https://etcher.io/> Acesso em: 10 Jan. 2018.

GAZETA DO POVO. **Automação residencial está mais acessível**. Disponível em: <http://www.gazetadopovo.com.br/imoveis/automacao-residencial-esta-mais-acessivel-351rpjeedgi5win9vowjei826>. Acesso em: 10 Abril. 2017.

HASSIO. **Instaling Hass.io**. Disponível em: <https://www.home-assistant.io/hassio/installation/>. Acesso em: 10 Jan. 2018.

HERRHOFRAT. **Add-on motion**. Disponível em: < https://github.com/HerrHofrat/hassio-addons>. Acesso em: 20 Fev. 2018.

HOMEASSISTANT. **Componente certificados**. Disponível em: < https://www.home-assistant.io/components/sensor.cert_expiry/>. Acesso em: 10 Jan. 2018.

HOMEASSISTANT. **Notification attachments**. Disponível em: < https://www.home-assistant.io/docs/ecosystem/ios/notifications/attachments/>. Acesso em: 10 Jan.

2018.

HOMEASSISTANT. **Notification sounds**. Disponível em: < https://www.home-assistant.io/docs/ecosystem/ios/notifications/sounds/>. Acesso em: 10 Jan. 2018.

HOMEASSISTANT. **Tutorial servidor web**. Disponível em: <https://www.home-assistant.io/docs/ecosystem/certificates/lets_encrypt/>. Acesso em: 10 Jan. 2018.

HOMEASSISTANT. **Android camera ip**. Disponível em: < https://www.home-assistant.io/components/android_ip_webcam/>. Acesso em: 10 Jan. 2018.

IHOUSE. **Catálogo IHOUSE.** Disponível em: < http://www.ihouse.com.br/>. Acesso em: 10 Maio. 2018.

LCSC. **Datasheet.** Disponível em: <https://datasheet.lcsc.com/szlcsc/TOSHIBA-ULN2004AFWG_C7513.pdf>. Acesso em: 10 Jan. 2018.

LONGO, Lucas. **Internet das coisas: uso de sensores e atuadores na automação de um protótipo residencial**. 100f. Monografia de Trabalho de Conclusão de Curso – Engenharia de Computação – Universidade Tecnológica Federal do Paraná. Pato Branco, 2015.

LUDWIG. Ricardo. **HouseManager - Comunicação sem fio de baixo custo para sistema de automação residencial**. 2015. 80 f. Monografia (Especialização em Redes de Computadores) - Universidade Tecnológica Federal do Paraná. Pato Branco, 2015.

PUSHBULLET. **Site do servidor**. Disponível em: < https://www.pushbullet.com >. Acesso em: 20 Fev. 2018.

PUSHSAFER. **Site do servidor**. Disponível em: < https://www.pushsafer.com >. Acesso em: 20 Fev. 2018.

ROBOCORE. **Catalogo Robocore.** Disponível em: < https://www.robocore.net/loja/produtos/sensor-de-presenca-pir-hc-sr501.html>. Acesso em: 20 Set. 2017.

SCHNEIDER e SOUZA. **Sistemas Embarcados: *Hardware* e Firmware na prática**. 2ª Edição. Editora Érica 2010 - 320p.

STAK. **Catalogo Stak.** Disponível em: < https://stak.com/Infrared_Remote_Control_module_Kit_HX1838_NEC_Code_VS1838B_Receiver>. Acesso em: 10 Jan. 2018.

TECONNECT. **Catalogo TE.** Disponível em: <http://www.te.com/usa-en/product-1-1419126-5.html>. Acesso em: 10 Jan. 2018.

TELECO. **Regulação Do Espectro De Radio Frequências: Análise Técnica Do Modelo Brasileiro. Disponível** em: <http://www.teleco.com.br/tutoriais/tutorialespecradio/default.asp>. Acesso em: 10

Abril. 2017.

W3TECHS. **Usage of server-side programming languages for *web*sites**. disponível em <http://w3techs.com/technologies/overview/programming_language/all>. Acesso: 10. Abril 2017.

WHATIS. **Whats my ip**. Disponível em: <http://www.whatsmyip.org/>. Acesso em: 10 Jan. 2018.

FONTES CONSULTADAS

AL-KUWARI, Ali Mohammed A. H.; ORTEGA-SANCHEZ, Cezar; SHARIF, Atif; POTDAR, Vidyasagar. **User Friendly Smart *Home* Infrastructure: BeeHouse**. In: 5th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2011), 31 May -3 June 2011, p. 257-262.

BRAGA. **Tudo Sobre Relés (Livro Completo)**. Disponível em: <http://www.newtoncbraga.com.br>. Acesso em: 10 Abril. 2017.

CARDOSO, F. R. M. e MACEDO-SOARES, J. C. T. **Método para implementação de redes sem fio**. Monografia de Graduação - Departamento de Engenharia Elétrica. Brasília: Universidade de Brasília, 2005 COIMBRA, T.R (2006).

DIAS, C. L. de A. **Domótica: aplicabilidade as edificações residenciais**. Dissertação - Universidade Federal Fluminense. Niterói, 2004.

ElShafee A., Hamed K. A. **Design and Implementation of a WiFi Based Home Automation System**. International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:6, No:8, 2012.

GOMEZ C., PARADELLS J. **Wireless *home* automation networks: A survey of architectures and technologies**. IEEE Communications Magazine.Volume: 48, Issue: 6. 2010.

HOMEASSISTANT. **Componentes**. Disponível em: <https://www.home-assistant.io/components/>. Acesso em: 10 Setembro. 2017.

HOMEASSISTANT. **Documentos GPS**. Disponível em: <https://www.home-assistant.io/getting-started/presence-detection/>. Acesso em: 10 Setembro. 2017.

HOMEASSISTANT. **Documentos groups**. Disponível em: <https://www.home-assistant.io/components/group/>. Acesso em: 10 Setembro. 2017.

HOMEASSISTANT. **Documentos IOS**. Disponível em: < https://www.home-assistant.io/docs/ecosystem/ios/notifications/basic >. Acesso em: 10 Setembro. 2017.

HOMEASSISTANT. **Documentos Templates**. Disponível em: < https://www.home-

assistant.io/docs/configuration/templating/>. Acesso em: 10 Setembro. 2017.

HOMEASSISTANT. **Documentos YAML**. Disponível em: <https://www.home-assistant.io/docs/configuration/yaml/ >. Acesso em: 10 Setembro. 2017.

HOMEASSISTANT. **Exemplos automação**. Disponível em: < https://www.home-assistant.io/docs/automation/examples >. Acesso em: 10 Setembro. 2017.

ICON. **Ícones usados.** Disponível em: < https://materialdesignicons.com/ >. Acesso em: 10 Setembro. 2017.

J. AUGUSTO, C. NUGENT. **Designing Smart homes: The role of artificial intelligence**. 1st ed. Ed. Springer. Berlin (Germany). 2006. pp. 1-8.

KO, Hoon; RAMOS, Carlos; **A Survey of context classification for intelligent systems research for Ambient Intelligence**. 2010 International Conference on Complex, Intelligentand *SOFTWARE* Intensive Systems. p.746-747.

MOZER, M. **The adaptive house. In: Intelligent Building Environments**, 2005. The IEESeminar on (Ref. No. 2005/11059). [S.l.: s.n.], 2005. p. 39-79. ISSN 0537-9989.

O'HARA, Bob; PETRICK, Al. **IEEE 802.11 handbook: a designer's companion**. 2. Ed. New York: IEEE, 2004. xxxvi, 364 p.

RAMOS, Amanda L. C. SANTOS, José Eduardo L. dos. **Sistema integrado de automação residencial com comunicação sem fio**. 2015. 67 f. Trabalho de Conclusão de Curso (Curso de Engenharia de Controle e Automação), Departamento Acadêmico de Eletrotécnica, Universidade Tecnológica Federal do Paraná. Curitiba, 2015.

SGARBI, Julio A., TONIDANDEL, Flavio. **Domótica Inteligente: Automação Residencial baseada em Comportamento**. In: Workshop de Teses e Dissertações em Inteligência Artificial, Ribeirão Preto, São Paulo, 2006.

SILVA, Denise S. da. **Desenvolvimento e Implantação de um Sistema de Supervisão e Controle Residencial**. 2009. 62 f. Dissertação (Mestrado em Engenharia Elétrica). Universidade Federal do Rio Grande do Norte, Natal-RN, 2009.

APÊNDICE A CODIGO DO ARQUIVO CONFIGURATION.YAML

```
1  http:
2    base_url:  https://<DOMINIO>.duckdns.org:8123  #  endereço
   padrão
3    ssl_certificate: /ssl/fullchain.pem  # valor fixo
4    ssl_key: /ssl/privkey.pem    # valor fixo
5    api_password: qwerty         # senha do site, opcional
6
7  panel_iframe:
8    configurator:
9      title: CONFIGURATOR        # nome do ícone no painel
10     icon: mdi:wrench           # ícone mdi
11     url:  https://<DOMINIO>.duckdns.org:3218  #  endereço  que  o
   add-on trabalha
12   terminal:                    # Referente ao add-on Terminal
13     title: Terminal
14     icon: mdi:console-network
15     url: https://<DOMINIO>.duckdns.org:7681
16
17 sensor:                        # lista de sensores
18   -  platform:  cert_expiry    #  tipo  de  sensor:  validade  d
   certificado
19     host: https:// <DOMINIO>.duckdns.org # endereço padrão
20     port: 8123                          # porta do endereço
21   - platform: time_date        # tipo de sensor: tempo e data
22     display_options:           # criar sensor e mostrar ícone
   na interface de usuário
23       - 'date'                 # criar sensor de apenas data
24       - 'time'               # criar sensor de apenas horário
25   - platform: history_stats # tipo de sensor: histórico de
   entidade, faz o somatório de horas no estado selecionado no
   período selecionado
26     name: Consumo mensal       # nome do sensor
27     entity_id: switch.condicionador  # entidade monitorada
28     state: 'on'                # estado da entidade
29     end: '{{ now() }}'         # termino na ultima hora
30     duration:                  # inicio 30 dias atrás
31       days: 30
```

```
32
33 binary_sensor:                  # lista de sensores binários
34   - platform: rpi_gpio         # tipo de sensor: input GPIO
35     ports:                     # pinos GPIO
36       17: PIR Sensor  # numero disponível e nome para o PGIO
37
38 switch:               # lista de saídas tipo chave
39   - platform: rpi_gpio
40     ports:
41       27: Sirene     # numero disponível e nome para o PGIO
42       23: Condicionador
43
44 camera:                      # lista de câmeras
45   - platform: local_file    # tipo de câmera: webcam
46     name: tcc_cam
47     file_path: /share/motion/lastsnap.jpg # local do arquivo
   aonde o stream de imagens é guardado
48
49 ios:                         # lista de comandos ios
50   push:                      # comando de notificação
51     categories:              # criação de categoria
52       - name: Alarme           # nome da categoria, irrelevante
53         identifier:   'alarm'   #  nome   da   identidade   da
   categoria, usado para chama-la
54         actions:
55           - identifier: 'SOUND_ALARM'    # nome da identidade
   do botão
56             title: 'Tocar Alarme'        # nome do botão
57             destructive: yes       #muda a cor do botão pra
   vermelho se 'yes'
58             activationMode:  'foreground'   # determina se o
   aplicativo irá abrir depois de apertado.
59           - identifier: 'SILENCE_ALARM'
60             title: 'Silenciar Alarme'    # nome do botão
61           - identifier: 'SHOW_FEED'
62             title: 'Mostrar Video'       # nome do botão
63
64 notify:                    # lista de serviços de notificação
65   - name: Android1          # nome da identidade do servidor
66     platform: pushsafer          #tipo de servidor pushsafer
67     private_key: CJKt0msACO4BRaZ4ONYk     #chave do servidor
68   - name: Android2          # nome da identidade do servidor
69     platform: pushbullet        #tipo de servidor pushbullet
70     api_key:  o.b4N713YJfxl25cL2ezfV3TtTATAnPUsA  #chave  do
   servidor
71
72 device_tracker:      # lista de serviços de rastreamento gps
73   - platform: gpslogger          #tipo de servidor gpslogger
74
```

```
75  zone:
76    - name: Home                        # nome da entidade da zona
77      latitude: -3.062560                           # latitude
78      longitude: -60.041699                         # longitude
79      radius: 250                       # raio em volta do local
80      icon: mdi:home-map-marker
81
82  input_number:                       # lista de entrada numérica
83    consumo:                                  # nome da entidade
84      name:  Potencia  media  do  condicionador  #  nome  na
    interface
85      icon: mdi:flash
86      min: 0                                    # valor mínimo
87      max: 6500                                 # valor máximo
88      step: 0.01                                   # numero inteiro
89      unit_of_measurement: kWh                  # unidade
90      mode: box        # entrada digitada, pode ser por slider
91
92  input_datetime:          # lista de entrada de data e horário
93    acionamento:                              # nome da entidade
94      name: acionamento do condicionador   # nome na interface
95      has_time: true                # entrada de horário apenas
96    desligamento:
97      name: desligamento do condicionador
98      has_time: true
99
100 recorder:                # gravar estado das entidades
101
102 history_graph:           # lista de gráficos
103   gr1:
104     entities:
105       - switch.condicionador
106     refresh: 3600        # atualização do gráfico em segundos
107     hours_to_show: 720   # horas para apresentar 24x30
108     name: Acionamento nos ultimos 30 dias
109
110 android_ip_webcam:
111   - host: 192.168.0.104 # Ip do celular
112     sensors:             # sensores para leitura
113       - battery_level
114       - battery_temp
115       - battery_voltage
116       - light
117       - motion
118       - pressure
119       - proximity
120       - sound
121     switches:             # atuadores para controle
122       - exposure_lock
```

```
      - ffc
      - focus
      - gps_active
      - night_vision
      - overlay
      - torch
      - whitebalance_lock
      - video_recording
```

APÊNDICE B CODIGO DO ARQUIVO AUTOMATION.YAML

```
  - alias: Notificar presenca IOS # Nome da automação/função
    initial_state:  false             # função desligada ao iniciar o sistema
    hide_entity: True                 # chave de desligamento da função escondida
    trigger:                          # Gatilho
      platform: state  # tipo de Gatilho: mudança de estado
      entity_id:    binary_sensor.pir_sensor    #    Entidade observada
      from: 'off'                     # De estado desligado
      to: 'on'                        # Para estado ligado
    action:                           # Ação
      service: notify.ios_iphonetcc  # Serviço notificar IOS
      data:                           # Dados do serviço
        title: "Atencao"              # Titulo da mensagem
        message: "selecionar opcao"  # Corpo da mensagem
        data:                    # Dados adicionais suportados pelo servidor de envio
          subtitle: "presenca detectada!"  # Subtítulo
          push:                 # Personalizações da mensagem
            sound:     "US-EN-Alexa-Motion-At-Front-Door.wav" # Som da mensagem
            category: "alarm"         # Categoria criada
          attachment:                 # Anexos
            url:           https://<DOMINIO>.duckdns.org:8123 # Endereço da interface de usuario

  - alias: Enviar video
    hide_entity: True
    trigger:
      platform: event               # tipo de Gatilho: evento
      event_type: ios.notification_action_fired # notificação de botão pressionado
      event_data:
        actionName: SHOW_FEED        # nome do botão
    action:
```

```
30        service: notify.ios_iphonetcc
31        data:
32          message: "stream da camera"
33          data:
34            attachment:
35              content-type: jpeg
36            push:
37              category: câmera  # categoria de envio de stream
38            entity_id: camera.ipcamera   # Entidade da câmera
39
40    - alias: Soar alarme
41      hide_entity: True
42      trigger:
43        platform: event
44        event_type: ios.notification_action_fired
45        event_data:
46          actionName: SOUND_ALARM
47      action:
48        - service: notify.ios_iphonetcc
49          data:
50            message: "Sirene ligada"
51        - service: switch.turn_on        # serviço ligar chave
52          entity_id: switch.sirene       # Entidade da chave
53
54
55    - alias: Desligar alarme
56      hide_entity: True
57      trigger:
58        platform: event
59        event_type: ios.notification_action_fired
60        event_data:
61          actionName: SILENCE_ALARM
62      action:
63        - service: notify.notify
64          data:
65            message: "Sirene desligada"
66        - service: switch.turn_off    # serviço desligar chave
67          entity_id: switch.sirene         # Entidade da chave
68
69    - alias: Notificar android pushsafer
70      initial_state: false
71      hide_entity: True
72      trigger:
73        platform: state
74        entity_id: binary_sensor.pir_sensor
75        from: 'off'
76        to: 'on'
77      action:
78        service: notify.Android1 # serviço notificar pushsafer
```

```
      data:
        title: Atencao
        message: >   # mensagem escrita em mais de uma linha
          [b][u][color=#FF0000] presença detectada
          [/color][/u][/b]       # Frase com estilo de letra
          [url=https://<DOMINIO>.duckdns.org:8123]Link da
          camera[/url]           # endereço da webcam
        data:
          icon: '25'# ícone enviado com a mensagem (1 a 176)
          iconcolor: "#FF0000"   # cor do ícone em HEXA
          sound: '25'            # som da mensagem (1 a 50)
          vibration: '3'   # intensidade da vibração (1 a 3)
          url: https:// <DOMINIO>.duckdns.org:8123
# endereço da interface de usuário
          urltitle: Webcam       # Nome do endereço
          time2live: '2'   # Tempo até mensagem ser deletada

  - alias: Notificar android pushbullet
    initial_state: false
    hide_entity: True
    trigger:
      platform: state
      entity_id: binary_sensor.pir_sensor
      from: 'off'
      to: 'on'
    action:
      service:    notify.Android2    #    serviço    notificar
pushbullet
      data:
        title: Atencao
        message: Presenca detectada
        data:
          url: https:// <DOMINIO>.duckdns.org:8123

  - alias: Lembrar de ativar vigilancia GPS
    trigger:
      - platform: zone            # tipo de Gatilho: zona
        entity_id:  device_tracker.gpstcc  # Entidade posição
gps do android
        zone: zone.home           # Zona casa
        event: leave       # evento posição gps sair da zona
      - platform: zone
        entity_id:    device_tracker.iphonetcc    #    Entidade
posição gps do IOS
        zone: zone.home
        event: leave
    action:
      - service: notify.ios_iphonetcc
        data:
```

```
122           title: "Lembrete"
123           message:    "Ativar    vigilancia    e    desligar
    condicionador"
124       - service: notify.Android1
125         data:
126           title: "Lembrete"
127           message:    "Ativar    vigilancia    e    desligar
    condicionador"
128       - service: notify.Android2
129         data:
130           title: "Lembrete"
131           message:    "Ativar    vigilancia    e    desligar
    condicionador"
132
133   - alias: Desligar condicionador automaticamente GPS
134     initial_state: false
135     trigger:
136       - platform: zone
137         entity_id: device_tracker.gpstcc
138         zone: zone.home
139         event: leave
140       - platform: zone
141         entity_id: device_tracker.iphonetcc
142         zone: zone.home
143         event: leave
144     action:
145       service: switch.turn_off      # serviço desligar chave
146       entity_id: switch.condicionador  # identidade da chave
147
148   - alias: Ligar sirene automaticamente
149     hide_entity: True
150     initial_state: false
151     trigger:
152       platform: state
153       entity_id: binary_sensor.pir_sensor
154       from: 'off'
155       to: 'on'
156     action:
157       - service: notify.ios_iphonetcc
158         data:
159           message: "Sirene ligada"
160       - service: notify.Android1
161         data:
162           message: "Sirene ligada"
163       - service: notify.Android2
164         data:
165           message: "Sirene ligada"
166       - service: switch.turn_on
167         entity_id: switch.sirene
```

```
168
169   - alias: Acionar condicionador horario
170     hide_entity: True
171     trigger:
172       platform: template       # Tipo de gatilho: Template
173       value_template: >        # Valor da template, notação
para escrever mais de uma linha
174         {{ states('sensor.time') == (states.input_datetime.
            acionamento.attributes.timestamp
175         | int | timestamp_custom('%H:%M', False)) }}
# se o horário do sensor 'time' for igual à entrada
'acionamento' convertida de timetamp para horas e minutos
176     action:
177       service: switch.turn_on
178       entity_id: switch.condicionador
179
180   - alias: Desligar condicionador horario
181     hide_entity: True
182     trigger:
183       platform: template
184       value_template: >
185         {{ states('sensor.time') == (states.
            input_datetime.desligamento.attributes.timestamp
186         | int | timestamp_custom('%H:%M', False)) }}
            # se o horário do sensor 'time' for igual à entrada
            'desligamento' convertida de timetamp para horas e
            minutos
187     action:
188       service: switch.turn_off
189       entity_id: switch.condicionador
190
191   - alias: Informacoes de consumo atual
192     hide_entity: True
193     trigger:
194       platform: time             # Tipo de gatilho: Tempo
195       minutes: '/30'
          seconds: 00                # a cada 30 minutos
196     action:
197       service: persistent_notification.create
# notificação na interface de usuário
198       data_template:             # Dados em template
199         title: "Informacoes de consumo atual"
200         notification_id: "1234" # numero de identificação da
mensagem, qualquer numero, para sobrescrever mensagens
antigas
201         message: >
202             O consumo do condicionado nos ultimos 30 dias,
            do mês 0{{ now().month | int - 1}}  ate
            # O consumo do condicionado nos ultimos 30 dias,
```

```
            do mês 0(mês anterior) ate
            {{ now().day | int - 1 }}/0{{ now().month}} é
            # (dia anterior)/0(mês atual) é
            {{ states('sensor.consumo_mensal')                |
            multiply( states('input_number.consumo') | float
            | multiply(0.033)) }} kWh,
            # (sensor dividido por 30 e multiplicado pelo
            consumo médio mensal) kWh,
            aproximadamente                                   {{
            states('sensor.consumo_mensal')    |    multiply(
            states('input_number.consumo')    |    float    |
            multiply(0.033) | multiply(0.806)) }}
            #  aproximadamente  (sensor  dividido  por  30,
            multiplicado  pelo  consumo  médio  mensal  e
            multiplicado pela tarifa atual sem tributos)
            R$  sem  tributos  {{"\n"}}  Atualizado  em  {{
            states('sensor.date')}}
            # R$ sem tributos, pulo de linha, Atualizado em
            (data atual)

  - alias: Consumo mensal
    hide_entity: True
    trigger:
      platform: time
      at: '12:00:00'
    condition:                              # condição extra
      condition: template                 # condição tipo template
      value_template: "{{ now().day == 1}}" # todo dia 1
    action:
      service: persistent_notification.create
      data_template:
        title: "Consumo mensal"
        message: >
            O consumo do condicionado nos ultimos 30 dias,
            do mes 0{{ now().month | int - 1}} ate
            # O consumo do condicionado nos ultimos 30 dias,
            do mês 0(mês anterior) ate
            {{ now().day | int - 1 }}/0{{ now().month}} é
            # (dia anterior)/0(mês atual) é
            {{ states('sensor.consumo_mensal')                |
            multiply( states('input_number.consumo') | float
            | multiply(0.033)) }} kWh,
            # (sensor dividido por 30 e multiplicado pelo
            consumo médio mensal) kWh,
            aproximadamente                                   {{
            states('sensor.consumo_mensal')    |    multiply(
            states('input_number.consumo')    |    float    |
            multiply(0.033) | multiply(0.806)) }}
            #  aproximadamente  (sensor  dividido  por  30,
```

```
                multiplicado pelo consumo médio mensal e
                multiplicado pela tarifa atual sem tributos)
225             R$ sem tributos {{"\n"}} Atualizado em {{
                states('sensor.date')}}
                # R$ sem tributos, pulo de linha, Atualizado em
                (data atual)
227
```

APÊNDICE C CODIGO DO ARQUIVO GROUPS.YAML

```yaml
  default_view:                    # substitui a aba principal
    view: yes                      # cria nova aba
    icon: mdi:collage
    control: hidden                # Chave geral de controle estado das entidades escondida
    entities:                      # Lista de entidades
      - group.Sistema_de_vigilancia
      - group.Controle_do_condicionador
      - binary_sensor.pir_sensor
      - sensor.date
      - sensor.time
      - sensor.tempo_humidity
      - sensor.tempo_symbol
      - sensor.tempo_temperature
      - sensor.consumo_mensal

  Sistema de vigilancia:           # Nome do grupo
    control: hidden                # Chave geral de controle estado das entidades escondida
    entities:                      # Lista de entidades
      - switch.sirene
      - camera.tcc_cam
      - automation.lembrar_de_ativar_vigilancia_gps # entidade criada pela automação do apêndice B
      - group.ativar_vigilancia_sirene_automatica # entidade criada pelo grupo na linha 22
      - group.ativar_vigilancia_sirene_manual     # entidade criada pelo grupo na linha 29
      - media_player.samsung_led32_un32j4303
      - camera.ip_webcam

  Controle do condicionador:
    control: hidden
    entities:
      - input_datetime.acionamento
      - input_datetime.desligamento
```

```
      - input_number.consumo
      - switch.condicionador
      - automation.desligar_condicionador_automaticamente_gps
      - history_graph.acionamento_nos_ultimos_30_dias
      - persistent_notification.1234

  Ativar vigilância Sirene automatica:
    entities:        # todas as entidades do grupo estão com atributos escondido logo o grupo fica escondido
      - automation.notificar_intruso_ios
      - automation.notificar_android_pushsafer
      - automation.notificar_android_pushbullet
      - automation.ligar_sirene_automaticamente

  Ativar vigilancia Sirene manual:
    entities:        # todas as entidades do grupo estão com atributos escondido logo o grupo fica escondido
      - automation.notificar_intruso_ios
      - automation.notificar_android_pushsafer
      - automation.notificar_android_pushbullet

  Android camera:
    view: yes
    icon: mdi:android-head
    control: hidden
    entities:
      - group.Ip_camera_1
      - sensor.ip_webcam_light_level
      - sensor.ip_webcam_motion
      - sensor.ip_webcam_sound
      - sensor.ip_webcam_battery_level
      - sensor.ip_webcam_battery_temperature
      - sensor.ip_webcam_proximity
      - sensor.ip_webcam_battery_voltage

  Ip camera 1:
    control: hidden
    entities:
      - camera.ip_webcam
      - switch.ip_webcam_exposure_lock
      - switch.ip_webcam_focus
      - switch.ip_webcam_frontfacing_camera
      - switch.ip_webcam_gps_active
      - switch.ip_webcam_night_vision
      - switch.ip_webcam_overlay
      - switch.ip_webcam_torch
      - switch.ip_webcam_video_recording
      - switch.ip_webcam_white_balance_lock
```