UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA
ENGENHARIA ELÉTRICA – HABILITAÇÃO EM ROBÓTICA E AUTOMAÇÃO INDUSTRIAL.

Douglas de Assis Ferreira

PROJETO DE UM SISTEMA DE SEGURANÇA E AUTOMAÇÃO RESIDENCIAL USANDO DE MENSAGENS VIA SMARTPHONE

Juiz de Fora
2016

Douglas de Assis Ferreira

PROJETO DE UM SISTEMA DE SEGURANÇA E AUTOMAÇÃO RESIDENCIAL USANDO DE MENSAGENS VIA SMARTPHONE

Monografia apresentada como avaliação parcial para obtenção do título de Engenheiro Eletricista da Universidade Federal de Juiz de Fora, submetida à aprovação da banca examinadora:

Orientador: D.Sc Leandro Rodrigues Manso Silva.

Aprovada por

BANCA EXAMINADORA

_____

Prof. Dr. Eng. Leandro Rodrigues Manso Silva - Orientador
Universidade Federal de Juiz de Fora

_____

Prof. D. Sc. Fabrício Pablo Virgínio de Campos
Universidade Federal de Juiz de Fora

_____

Eng. Renato Ribeiro Aleixo
Universidade Federal de Juiz de Fora

Juiz de Fora

2016

AGRADECIMENTOS

Primeiramente agradeço aos meus pais Sebastião Melquior Ferreira e Edimea das Graças Fiochi de Assis Ferreira, por todo o apoio incondicional e conselhos ao longo dessa árdua jornada.

Gostaria também de agradecer aos meus irmãos, Nicolas e Lays, por sempre se preocuparem com o meu bem-estar e fazerem o que fosse possível por mim.

À minha namorada, Thaís, pela compreensão, amor, carinho e apoio dado nos meus piores momentos.

Aos professores e amigos Leandro Manso, Manuel Rendón, Cristiano Casagrande, Leonardo Olivi, Thiago Coelho e Exuperry Bastos, por sempre me passarem seus conhecimentos, experiências e sabedorias de vida durante todo tempo que passamos juntos.

Aos meus amigos do Cubo Mágico, muito obrigado por sempre me ajudarem a testar o meu projeto, além de serem pacientes com toda bagunça que faço.

Aos amigos que fiz na graduação, em especial Felipe Ferraz, Daniel Ramalho e Thiago Trindade: trabalhar com vocês me trouxe outra perspectiva que, juntos, nós podemos sempre; e ao Marcello Guedes e Leandro Ribeiro, amigos que sempre andaram junto comigo nessa longa jornada.

Aos meus familiares sempre presentes pelo incentivo e torcida.

Ao Laboratório de Eficiência Energética, LENNER e aos que lá trabalham.

À Universidade Federal de Juiz de Fora por toda a estrutura disponibilizada para nosso curso, e a todos os professores que se esforçaram diariamente para que o conhecimento fosse por mim adquirido.

Muito obrigado a todos!

RESUMO

O presente trabalho apresenta um protótipo de sistema de segurança e automatização residencial usando sistema embarcado, onde tudo é controlado por um trocador de mensagem.

As implementações foram feitas no Microsoft Visual Studio e passada para a placa usando o WINSCP, programa cliente SFTP e FTP. Foi feita dessa forma à fim de agilizar a implementação, devido à toda dificuldade de se programar em linhas de comando.

O sistema é todo controlado a partir de trocas mensagens do usuário com a casa a partir do aplicativo Telegram.

Para fazer o circuito, usou-se relés digitais de 10 A, protoboard, jumpers, uma webcam e uma lâmpada.

Palavras-chave: 1. Automatização. 2. Telegram 3. Acionamento.

SUMÁRIO

Sumário

Sumário de figuras

1. INTRODUÇÃO

1.1. CONSIDERAÇÕES INICIAIS

A humanidade sempre pesquisou meios de tornar processos mais produtivos, mais ágeis e mais seguros. Vezes esses avanços foram lentos, vezes muito rápidos, como nos últimos 50 anos, onde centros de pesquisas se multiplicaram mundo à fora para melhorarem rendimentos bélicos e industriais e posteriormente, adaptados para o restante da sociedade. Vide GPS, satélites de telecomunicações, computadores, forno micro-ondas, entre outros.

Atualmente, estamos vivendo uma nova revolução industrial, a quarta de toda história. Nessa, os responsáveis pela radical mudança serão os robôs e os sistemas interligados em uma rede inteligente, onde todos os equipamentos se comunicarão entre si. Em outras palavras, os equipamentos terão acesso à internet para que possam trocar informações, automatizando assim o controle de diversos processos.

Assim como ocorreu no passado com os computadores, onde eles eram fabricados somente para fins de pesquisa e industriais devido ao seu alto custo, equipamentos inteligentes poderão começar a ser utilizados para proporcionar conforto, segurança e praticidade no dia-a-dia de todos. A utilização desses equipamentos presentes em uma rede doméstica possibilita a troca de informações entre eles, assim, surgindo a automação residencial, também conhecida como ***Domótica*** (junção da palavra latina “Domus” (casa) com Robótica).

Conceitos como casas e prédios inteligentes estão sendo definidos pela aplicação de mecanismos automáticos, que se comuniquem entre si, e possam também ser controlados remotamente por um usuário. Dentro desse contexto, o presente trabalho propõe a implementação de um sistema de automação e segurança residencial, controlado por uma plataforma Linux Embarcado e que utiliza um aplicativo de troca de mensagens, capaz de comunicar-se através da internet com computadores e/ou smartphones.

1.2. Objetivos

O objetivo do presente trabalho é criar um protótipo que possa possibilitar um projeto elétrico inteligente que possibilite o acesso à terminais elétricos (tomadas, lâmpadas e câmeras) de maneira remota usando um trocador de mensagem.

1.3.ESTRUTURA DO TRABALHO

O atual trabalho foi subdividido em dez capítulos descritos a seguir:

O Capítulo 2 apresenta uma revisão a respeito dos conceitos de automação residencial, bem como alguns equipamentos para esse fim.

O Capítulo 3 apresenta o conceito do dispositivo proposto, e descreve as partes que o compõe.

O Capítulo 4 descreve a implementação de cada parte constituinte do dispositivo de automação e segurança residencial proposto.

O Capítulo 5 apresenta o comportamento do dispositivo perante algumas situações de operação.

Por fim, o Capítulo 6 apresenta as conclusões a respeito deste trabalho e as propostas para trabalhos futuros.

2. Automação Residencial - Domótica

Domótica é uma tecnologia recente e é responsável pela gestão de todos os recursos de uma residência. Este termo nasceu da fusão da palavra “Domus”, que significa casa, com a palavra “Robótica”, que está ligada ao ato de automatizar, isto é, realizar ações de forma automática. [1]

Com a domótica, não só os telefones celulares, PDAs e notebooks funcionarão nas novas redes de comunicação. Geladeiras, TVs de alta resolução, fornos de microondas, câmeras digitais etc., possuirão conexões em rede, permitindo seu controle e monitoramento à distância (PINHEIRO, 2004). [2]

Ela, a domótica, utiliza as vantagens de meios eletrônicos e de informação para proporcionar um gerenciamento de equipamentos presentes em meios residenciais e comerciais, cujo o sistema pode ser operado de maneira passiva (reagindo à ordens enviadas pelo operador) ou de maneira automática, interpretando sinais vindo de sensores e atuando sem necessidade de intervenção humana [3].

A operação automática do sistema possui duas arquiteturas, a ABA (Arquitetura Baseada em Automação), onde dispositivos como controles remotos, sensores de presença, sensores de movimentos, dispositivos biométricos e entre outros, são ajustados e configurados pelos usuários de acordo com as suas necessidades, já a ABC (Arquitetura Baseada em Comportamento), evolução da ABA, é chamada de “Domótica Inteligente”, onde o sistema possui vários bancos de dados para armazenar ações de atuadores e leituras de sensores para poder avaliar os dados à fim de adaptar as regras de automação do ambiente aos habitantes do recinto [4], pois os seres humanos estão em constante mudança; o que é uma regra ou rotina hoje, amanhã pode não ser. Os hábitos, horários e atividades mudam com o passar do tempo e os sistemas têm de aprender e se adaptar às mudanças de seus usuários [5]. Em outras palavras, a Domótica Inteligente possui características fundamentais de um sistema inteligente: ter memória; ter noção temporal; fácil interação com os habitantes; capacidade de integrar todos os sistemas do ambiente; atuar em várias condições; facilidade de reprogramação e capacidade de autocorreção [1].

Para dar tal inteligência e “consciência”, vários algoritmos estão sendo desenvolvidos, como o ID3, [6], técnica de aprendizado que consiste na indução de uma descrição geral a partir de um conjunto de exemplos, chamado de “Conjunto de Treinamento” [4], e o C4.5, algoritmo

mais desenvolvido que permite trabalhar com atributos de valores contínuos e até mesmo desconhecidos. [5]

As atuais pesquisas na área de automação predial se concentram, em sua maioria, na rede de comunicação entre de sensores e atuadores, nos protocolos usados nessa comunicação e ainda em sistemas de gerenciamento e otimização do uso e aproveitamento de recursos [7]

A diversidade de sistemas, produtos e protocolos existentes (e em desenvolvimento) para automação residencial no mundo é grande, criando uma imensa dificuldade de integração de produtos voltados para a área, fazendo as grandes empresas se unirem para buscar soluções que facilitasse a instalação e comunicação de diferentes equipamentos para os mais diversos fabricantes, gerando assim consórcios e associações que juntas, criaram várias tecnologias domóticos, esses podendo ser por condutores definidos ou sem fio. [8]

Os sistemas com tecnologia de condutores definidos mais presentes no mundo são o IHC (Inteligence House Control), lonworks, EIB (European Installation Bus) e a X10 - PLC (Power Line Carrier), este último promissor devido à possibilidade de envio de informação pela instalação elétrica. [8]

O outro tipo de sistema citado é o *Wireless*, onde os padrões mais usados são o Wi-Fi, Bluetooth e o WiMax, mas há também o ZigBee e o Z-Wave, cujo primeiro é uma tecnologia criada pela *Aliança ZigBee,* formanda pela Honeywell, Invensys, Mitsubishi, Motorola e Samsung, cujo objetivo foi desenvolver potenciais facilidades através de uma plataforma que proporcionasse controle e monitoramento remoto, de forma simples, confiável, de baixo custo e reduzido consumo de energia [8] enquanto o segundo utiliza tecnologia de transmissão em radiofrequência para comunicação, onde a sua estrutura é formada por malhas de “nós”, cuja a informação vai de um ponto ao outro do sistema passando por um ou mais “nós”, dando robustez e confiabilidade do recebimento da mensagem. [Sylvania Home Automation (2005)] [9]

Outro problema encontrado é o custo dos projetos para a concepção das residências inteligentes, onde somente proprietários de maior poder aquisitivo ou grandes empreendimentos conseguem usufruir das vantagens da domótica. Entretanto, com o passar do tempo, a automação residencial deve fazer parte da rotina doméstica da classe média, como ocorreu com os computadores, internet, freezer etc.[8]

3. Proposta de um dispositivo de automação e segurança residencial

No presente trabalho é proposto um dispositivo de segurança e automação residencial. Esse dispositivo é formado por diversos componentes, entre eles: uma central de processamento, sensores, atuadores e uma interface homem-máquina (IHM). Esses componentes interagem de acordo com o diagrama mostrado na Figura 1.

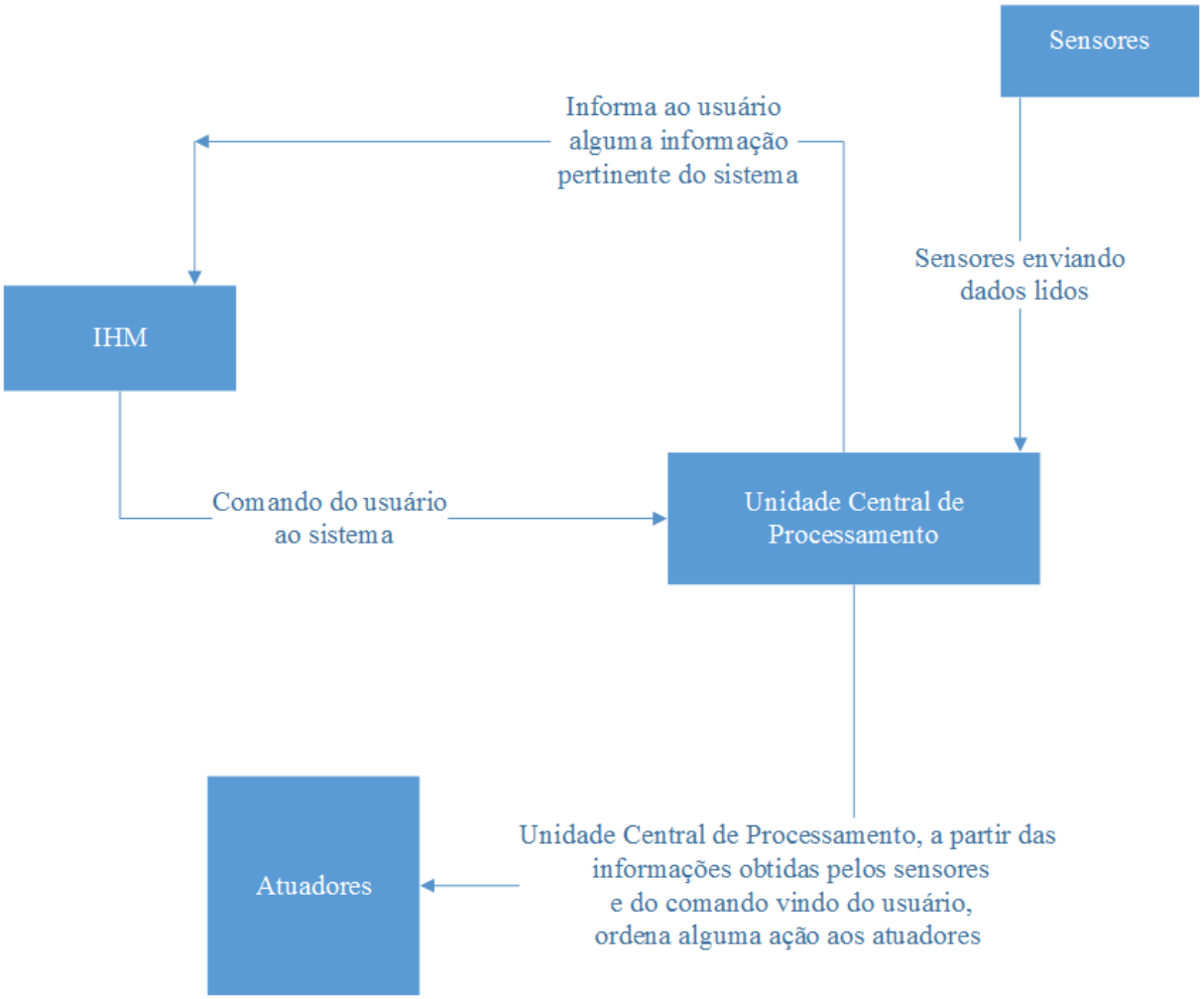

Figura 1 - Representação do sistema

A central de processamento é constituída pelo módulo de processamento Intel Edison, e é responsável por realizar todo o gerenciamento dos outros componentes, realizando tarefas como: receber dados dos sensores, controlar a ação dos atuadores e trocar mensagens com a IHM afim de comunicar algum evento, ou executar alguma ação requerida pelo usuário.

Como sensores, este sistema conta com um sensor ultrassônico que funcionará como um sensor de presença e uma câmera com comunicação usb para registro de imagens, ambos farão parte do sistema de segurança implementado. Como atuadores, o sistema conta com relés que serão utilizados para acionar cargas residenciais como lâmpada e ventilador. Essas cargas serão acionadas de acordo com as mensagens enviadas através da IHM.

A IHM é constituída pelo aplicativo de troca de mensagens Telegram [10], que estará embarcada na plataforma Intel Edison e poderá se comunicar com o aplicativo em smartphones ou computadores de usuários cadastrados no sistema. Nas seções a seguir, os componentes que constituem o sistema serão descritos.

3.1.Módulo Intel Edison

A Intel Edison é um módulo de processamento, criado pela Intel para o mundo embarcado. Ela é um minicomputador de baixo consumo que possui uma série de recursos de processamento e comunicação importantes que garantem a ela a habilidade necessária para movimentar os mais diversos tipos de *gadgets* e aparelhos eletrônicos. A Figura 2 mostra uma foto desse módulo.

Figura 2 – Módulo Intel Edison. [10]

O processador residente nessa placa é o dual-core Quark SOC, que opera a 400 MHz, possui memória RAM LPDDR2, integrada ao mesmo chip do processador (SOC) e uma NAND Flash para armazenamento. Também oferece diversos IOs para expansão, incluindo barramentos de comunicação I2C, I2S, UART, SPI, além de permitir controle PWM. O processador é um chip de diversas camadas. A segunda camada carrega os controladores WiFi e Bluetooth, que permite que a placa, mesmo bem pequena, se conecte aos aparelhos sem fios ao seu redor. Na face de trás do módulo Edison estão alguns conectores para que o Edison possa se conectar com outros dispositivos, para isso, basta conectá-lo a uma placa auxiliar que contenha todos os outros componentes que serão controlados pelo Edison. O dispositivo também possui 512 mb de memória DRAM integrada, e de 1 a 2 GB para armazenamento.

O sistema operacional que controla o Edison é uma versão simplificada do Linux, o Yocto. Uma das principais vantagens dele é o consumo de energia: o máximo de energia utilizada pelo Edison é 1 Watt, mas o consumo médio fica na casa dos 250 mW, sendo que, para tarefas mais simples, ele pode acabar utilizando ainda menos energia. A Figura 3 nos mostra os detalhes da placa Intel Edison.

A Figura 3 dois mostra a placa que a Intel Edison controla com as suas saídas digitais e analógicas, PWM e outros.

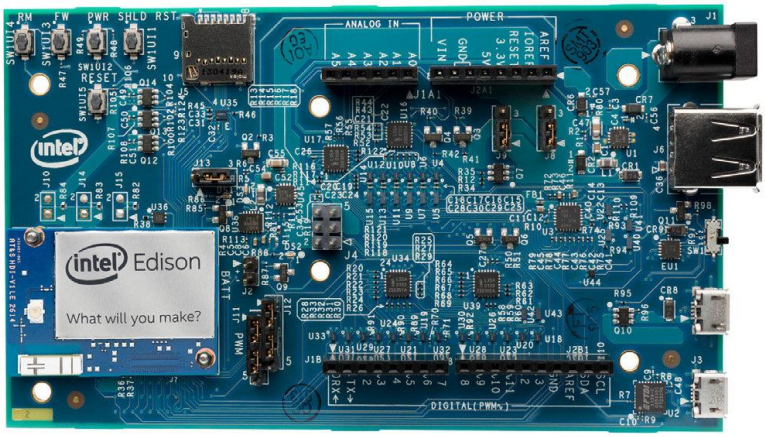

Figura 3 - Shield para a Intel Edison [10]

3.2.Módulo Relés

Para o acionamento das cargas, foi utilizado um módulo que contém dois relés de 5V/10A, SRD-05VDC-SL-C, este módulo é facilmente encontrado no mercado e pode ser utilizado com qualquer circuito digital ou processador como: Arduino, Rapsbery Pi, entre outros [2]. A Figura 4 mostra uma foto desse módulo.

Figura 4 - Relé arduino [2]

Cada relé desse módulo suporta cargas de até 10 A, em 125 VAC, 250 VAC ou 30 VDC. Leds indicadores mostram o estado do relé (ligado/desligado) em cada canal. O módulo já contém todo o circuito de proteção para evitar danos ao micro controlador, e possui baixa corrente de operação. [2]

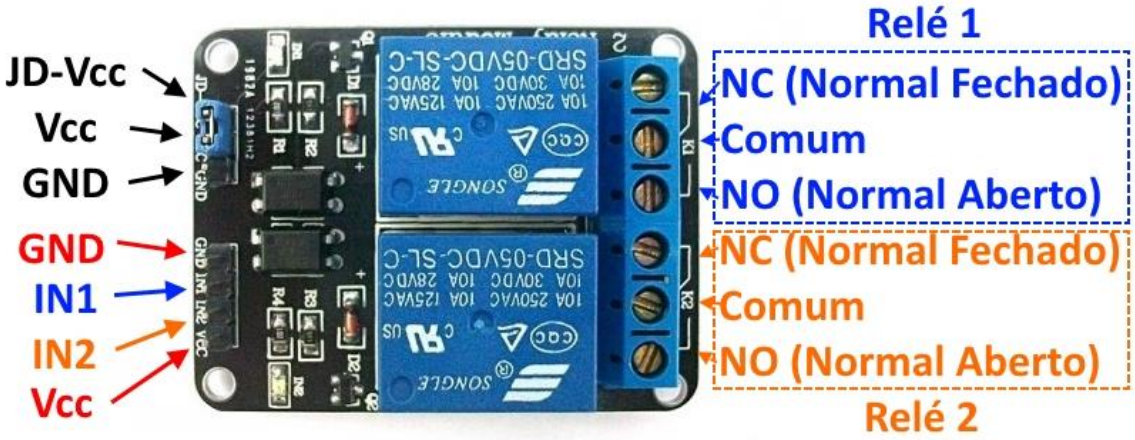

Figura 5 - Esquema da pinagem do rele SRD-05VDC-SL-C [2]

Na Figura 6 é possível observar a pinagem do módulo relé SRD-05VDC-SL-C. No lado esquerdo superior os pinos JD-Vcc, Vcc e GND, que permitem que seja conectada uma fonte externa de 5V. Abaixo, os pinos GND, IN1 (aciona o relé 1), IN2 (aciona o relé 2), e o Vcc. Ao lado dos relés, os contatos NC (Normal Fechado), C (Comum), e NA (normal aberto)

3.3.Sensor de distância ultrassônico

Para determinar presença ou movimento no ambiente, utilizou-se o sensor HC-SR04, que é capaz de medir distâncias de 2cm a 4m com ótima precisão e baixo custo. Este módulo possui um circuito com emissor e receptor acoplados e 4 pinos (VCC, Trigger, ECHO, GND) para medição, ilustrado na Figura 6

Figura 6 - Sensor ultrassônico HC-SR04. [11]

O funcionamento do HC-SR04 [11] se baseia no envio de sinais ultrassônicos pelo sensor, que aguarda o retorno (**echo**) do sinal, e com base no tempo entre envio e retorno, calcula a distância entre o sensor e o objeto detectado. [11]

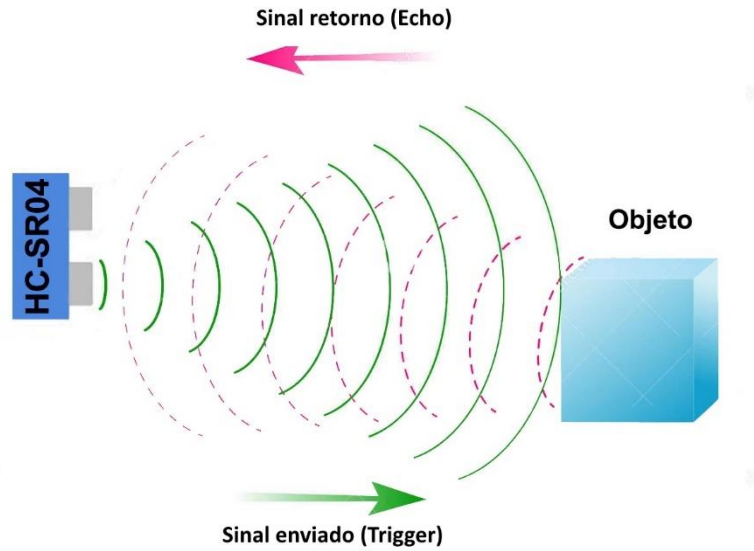

Figura 7 - Ilustração de funcionamento do HC SR04 [11]

Primeiramente é enviado um pulso de 10µs, indicando o início da transmissão de dados. Depois disso, são enviados 8 pulsos de 40 KHz e o sensor então aguarda o retorno (em nível alto/high), para determinar a distância entre o sensor e o objeto, utilizando a equação ***Distância = (Tempo echo em nível alto * velocidade do som) /2.*** [11]

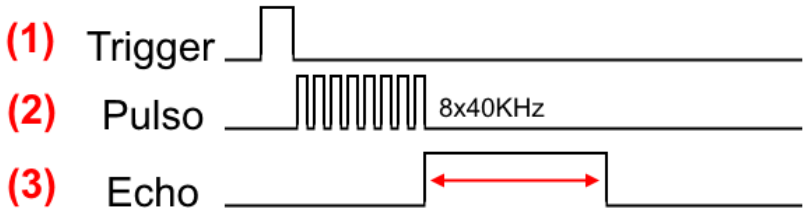

Figura 8 - Ilustração de funcionamento do HC SR04 [11]

Logo, ao pôr o sensor em um local fixo, a distância medida por ele será sempre a mesma até que algum objeto passe por ele.

4. Implementação do módulo de automação e segurança residencial.

Após a explicação de cada módulo de hardware que foi utilizado neste projeto, no Capítulo 3, este capítulo irá descrever como cada uma das funcionalidades foi implementada na plataforma de processamento Intel Edison. A linguagem de programação escolhida para a implementação foi a linguagem Python, devido à variedade de bibliotecas disponíveis para se trabalhar com Linux embarcado e com o módulo Intel Edison.

4.1. Telegram

A escolha pela utilização do Telegram foi fundamentada no fato de que o mesmo possui uma API (*Application Programming Interface*) aberta e bem documentada, possibilitando sua fácil integração com linguagens como Python, Java e C#. Além disso, o Telegram já possui uma estrutura específica para trabalhar com bots. [10]

4.1.1. Criando um Bot

Um bot é basicamente uma aplicação capaz de simular ações humanas, e até mesmo interagir por meio de si1stemas de mensagens instantâneas, tal como Telegram, e assim responder a perguntas, comandos, e desempenhar ações específicas. O interessante da estrutura do Telegram para bots é que eles não dependem de números de celular para operarem. Basicamente você realiza um processo de registro, obtém uma chave de acesso e utiliza essa chave para controlar o seu bot.

Todo o mecanismo de criação e gestão de bots em Telegram pode ser visto em detalhes na área de bots do Telegram [12]. Inclusive, eles mesmos já fornecem toda uma base de códigos-fontes com exemplos para interação com bots Telegram em diversas linguagens. Para criar um bot no Telegram, é preciso interagir com o "*The Botfather*"!

Outro fato interessante sobre o Telegram é que o mesmo não tem limitações quanto à forma de acesso. Com isso, pode-se fazer todo o setup do bot conversando com o *The Botfather* pelo navegador, no computador. Para tal, deve-se seguir os seguintes passos:

1) Acessa-se o link tg://resolve?domain=botfather, logados no Telegram e inicia-se o processo com o comando /start.
2) Para criar o boot digita-se o comando /newbot e aguarda-se ele pedir o nome do bot.
3) Após informa-lo o nome do bot, ele requisita um username para o bot que deve finalizar com a palavra bot.

4) Logo após o passo anterior, recebemos um token, que é uma chave de acesso para fazer interface com o nosso bot.

4.1.2. Configurar a Edison para Python com Telegram

A escolha pelo Python se deu pelo fato da simplicidade da linguagem, sendo objetiva, direta e intuitiva na grande parte das vezes.

Com a placa já conectada à internet, deve-se instalar o python com a versão 3.4.3 para conseguirmos usar a biblioteca do telegram.

Há um tutorial no apêndice para a instalação dessa versão do python. link abaixo se encontra o passo a passo para fazer, a instalação do programa e das bibliotecas necessárias usando o terminal.

4.1.3. Utilizando o Bot criado

Antes de se poder utilizar o Bot criado, deve configurar a Edison, instalando a versão correta do Python e também as bibliotecas necessárias para se utilizar a API do Telegram. Os passos necessários para tal estão mostrados no Apêndice A.

Após o configurar o Python na Edison e obter o Token do bot, para utilizar o bot num programa em Python, é necessário importar a biblioteca denominada “telegram”, e criar um objeto bot utilizando o Token obtido durante a criação do bot, utilizando a seguinte linha de código:

```
bot = telegram.Bot('266707858:AAETwfO6juMgM3_WnJJt2Fvf0tKbGOrun2c')
```

Esse objeto “bot”, pode ser utilizado para, por exemplo, realizar o envio de mensagens via telegrama para um usuário já cadastrado. Para isso, basta utilizar a seguinte função:

```
bot.sendMessage(chat_id=CHAT_id, text="Bot Iniciado.")
```

Em que parâmetro chat_id é um número que identifica cada usuário do telegrama e é obtido em sua primeira iteração com o bot.

4.1.4. Cadastro de Usuários

Para cadastrar um usuário, deve-se obter o seu chat_id, que é uma sequência de números que registra cada usuário do telegram. Com esse dado, consegue-se receber e enviar mensagens utilizando o bot criado. Dessa maneira, o chat via telegram se torna a IHM do sistema proposto.

Para conseguir tal registro, implementa-se um código onde o bot responde somente o id do remetente da mensagem como podemos ver na Figura 9. Com esse id em mãos, cadastra-se o usuário que irá operar e se comunicar com a casa.

Figura 9– Obtenção do id do usuário

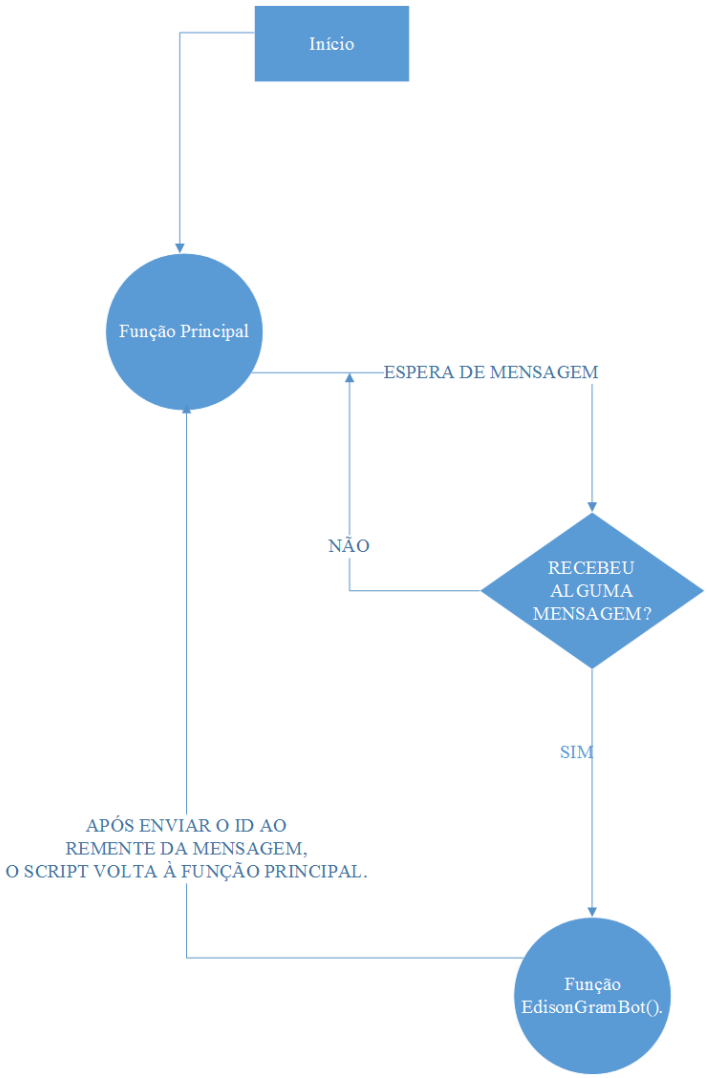

Figura 10 - lógica de implementação para obtenção do ID de remetentes de mensagem.

O processo ilustrado Figura 10, se inicia com a importação da biblioteca do telegram e com a definição do token fornecido pelo BotFather. Após, o processo entra na função principal, onde está presente uma variável que armazena o valor da última mensagem enviada ao bot (todas as mensagens são numeradas) e fica esperando uma mensagem ser enviada por algum usuário. Quando o bot recebe alguma mensagem, entra na função EdisonGramBot(), que armazena o id do remetente e responde como forma de mensagem o seu registro (id).

4.2. Obtendo arquivos de imagem e enviando-o pelo Telegram.

A placa que está acoplada da Intel Edison possui uma entrada USB, onde se conectará a webcam, como podemos ver na Figura 11.

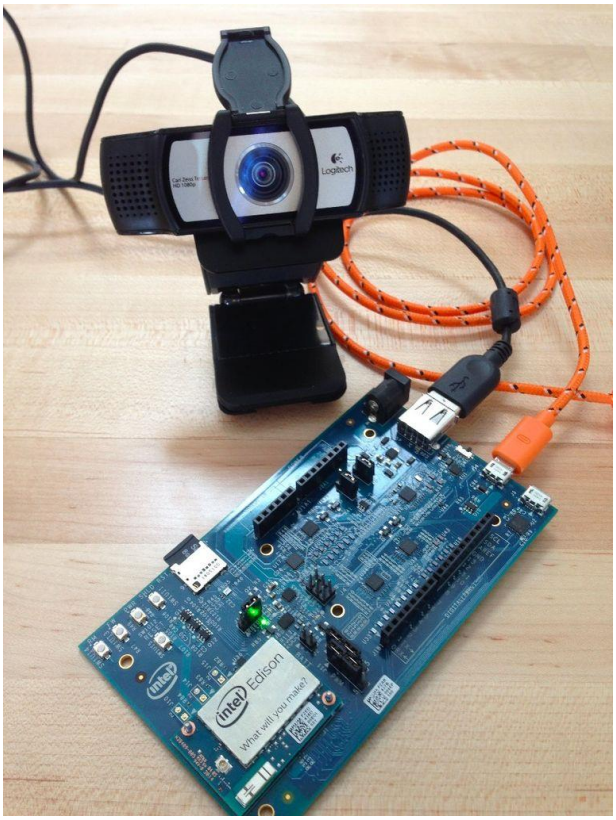

Figura 11- Conexão da Intel Edison com a webcam utilizada

Para a utilização da webcam, foi necessário a instalação do seu respectivo driver, como está mostrado no Apêndice A. Para obter os arquivos de imagem junto a câmera, é necessário o seguinte comando no terminal do Linux embarcado na placa.

```
./ffmpeg -s 320x240 -f video4linux2 -i /dev/video0 -vframes 1 image.jpeg
```

Com isso, a imagem será capturada, será salva no diretório em que está contido a aplicação “ffmpeg” com o nome de image.jpg. Para obter a imagem utilizando o programa escrito em Python, pode-se fazer uso da biblioteca “os”, que emula o terminal de comandos. Para isto, basta utilizar a seguinte linha de código.

```
os.system ("./ffmpeg -s 320x240 -f video4linux2 -i /dev/video0 -vframes 1 image.jpeg")
```

Para enviar essa imagem pelo telegram utilizando o bot, usa-se o comando:

```
bot.sendPhoto(chat_id=administrador, photo=open('image.jpeg', 'rb'))
```

em que, administrador é uma variável que contém o id do usuário que irá receber a imagem do arquivo image.jpeg.

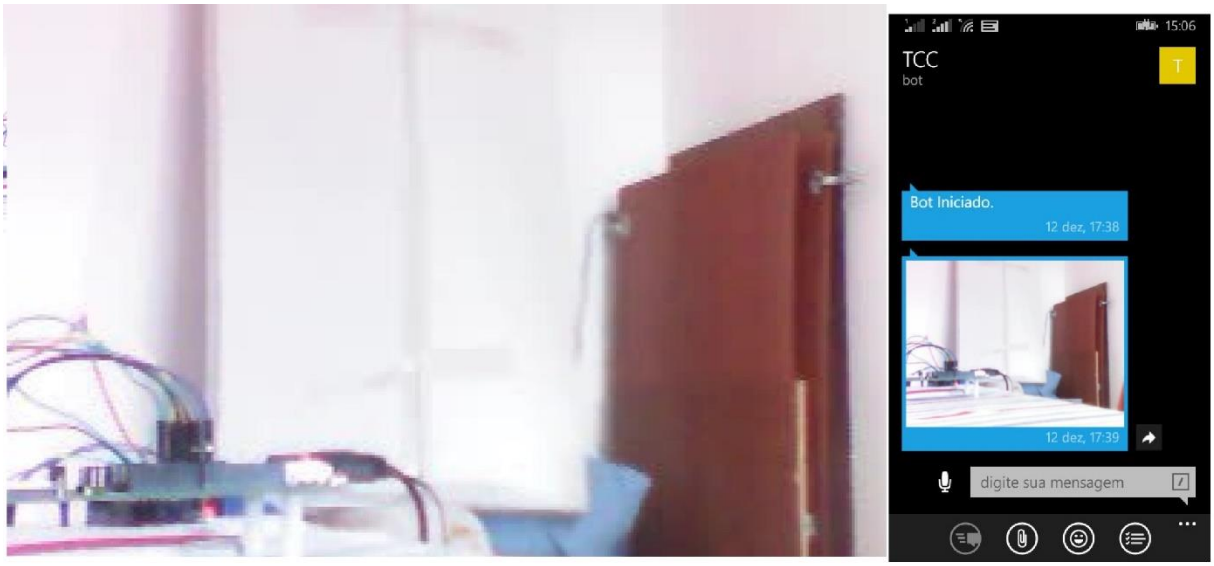

Figura 12 - Imagem enviada pelo bot através do Telegram

Na Figura 12, é mostrada do lado esquerdo a imagem obtida com a webcam, e no lado direito, a tela do telegram do usuário que recebeu a mensagem.

4.3. Implementando a lógica do Sensor de Presença

O sensor de presença é utilizado para constituir, em conjunto com a câmera, o sistema de segurança. Quando a presença é detectada, uma foto é capturada e enviada via telegram para o usuário.

O sensor utilizado, mede a distância do mesmo até um objeto. Sendo assim, a presença será detectada assim que a distância medida for menor ou igual a um valor pré-determinado.

A lógica utilizada para implementar um sensor de presença usando o sensor HC-SR04 é bem simples: Primeiro, é necessário definir uma distância que não indique presença, como por exemplo distância do sensor até a parede. Assim que uma porta abrir, a distância pode diminuir ou aumentar e essa mudança indicará que houve algum evento que deva ser notificado ao usuário do sistema, avisando por uma mensagem contendo a imagem do ambiente ou da pessoa presente no mesmo. A lógica implementada está mostrada no fluxograma da Figura 13.

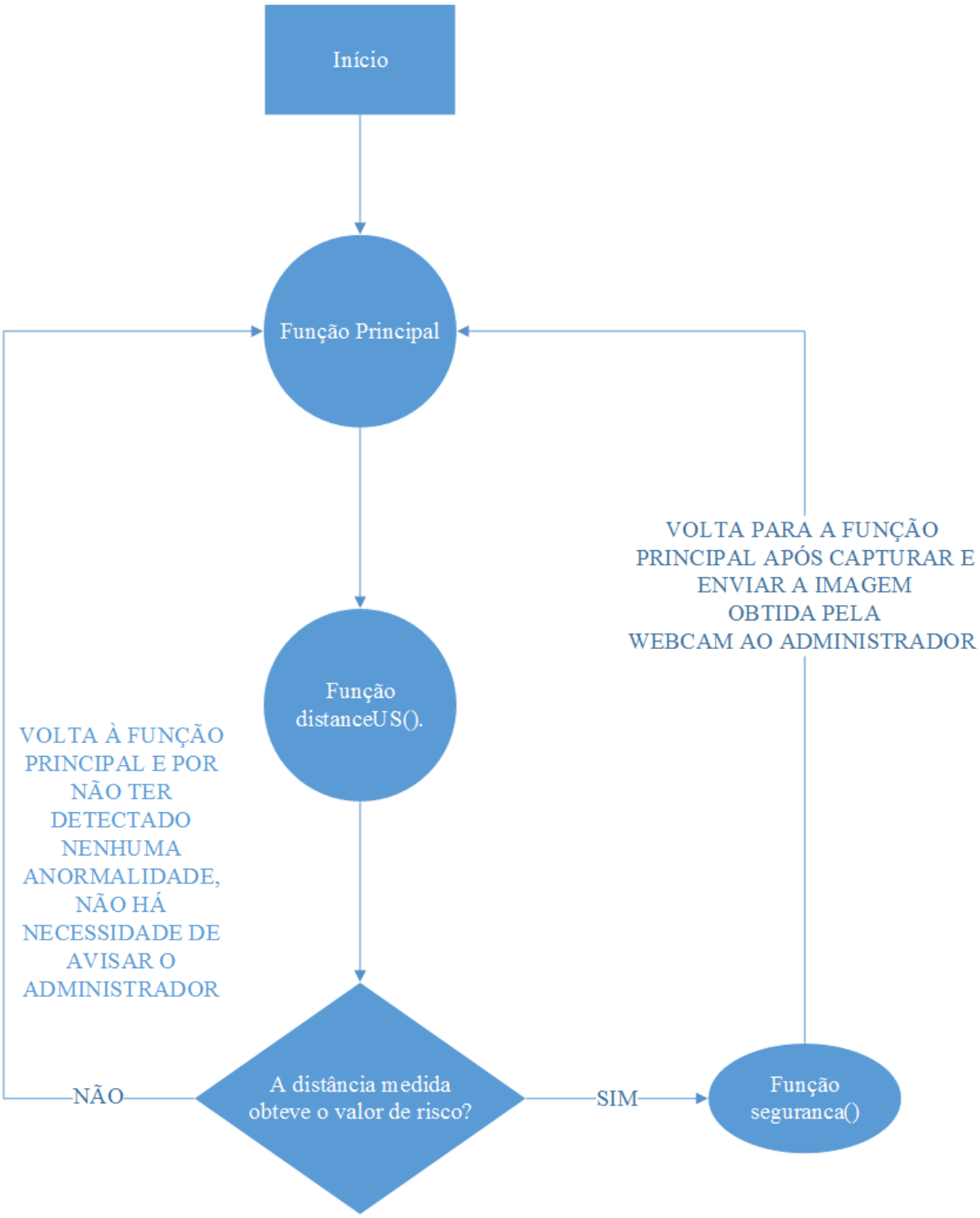

Figura 13 - Lógica do detector de presença usando um sensor HC-SR04

A Figura 14 mostra a imagem recebida pelo usuário quando uma presença é detectada.

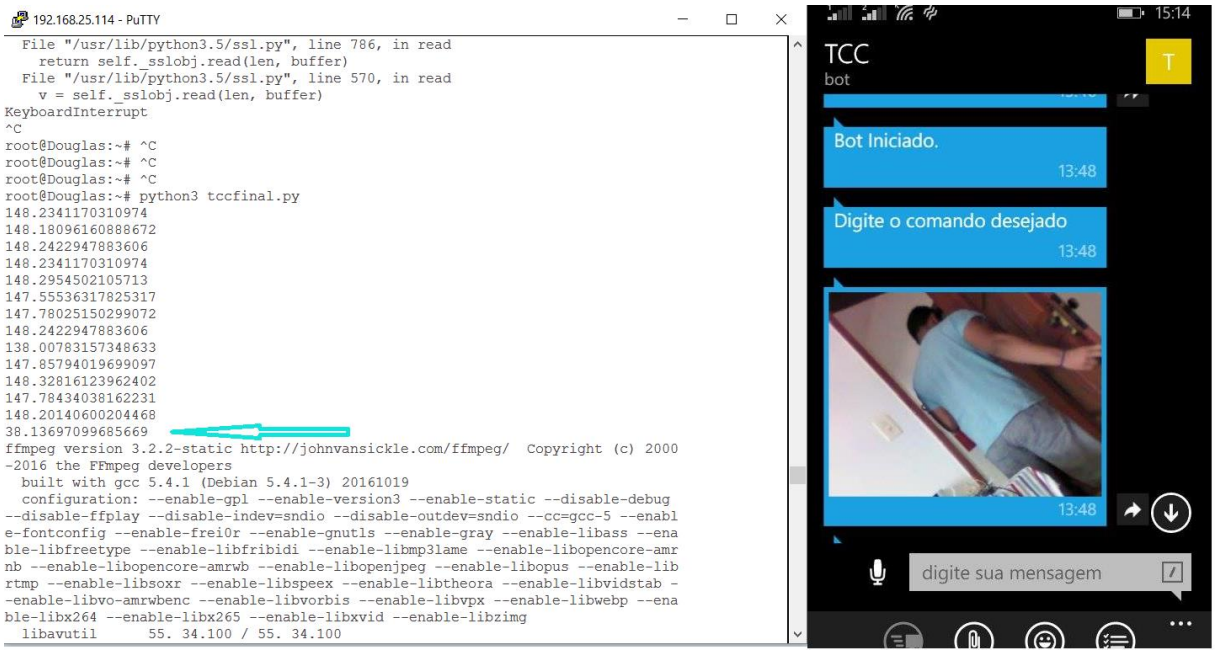

Figura 14 - Detecção de movimento e/ou presença e envio da imagem ao administrador/usuário.

No lado esquerdo da Figura 14encontra-se o terminal do Linux Embarcado presente na Intel Edison com as leituras do sensor HC-SR04 e do disparo de captura de imagem. Observa-se que a partir do momento que o sensor mediu uma distância menor que 100 cm, no caso foi 38.13cm, iniciou-se a captura da imagem que por sua vez foi enviada ao usuário, mostrado no lado direito.

4.4.Acionamento de relés.

O acionamento de cargas é dado por controle de relé através de comandos do usuário, que dependendo da mensagem, gerará uma determinada ação. Um exemplo seria o acender de uma lâmpada. Para acionar os reles, é necessário importar a biblioteca de I/O “mraa”, configurando corretamente os pinos em que as cargas estarão conectadas. Por exemplo, conectando a lâmpada no GPIO 7, basta utilizar o código mostrado na Figura 15 para configurar a utilização da biblioteca “mraa”.

```
lampada = mraa.Gpio(7)      # atribuição do I/O da lâmpada
lampada.dir(mraa.DIR_OUT)   # configuração do I/O como saída
lampada.write(1)            # coloca o I/O em nível lógico alto
```

Figura 15 – Código para configuração do I/O da lâmpada.

A lógica implementada para o acionamento das cargas está descrita pelo fluxograma da Figura 16.

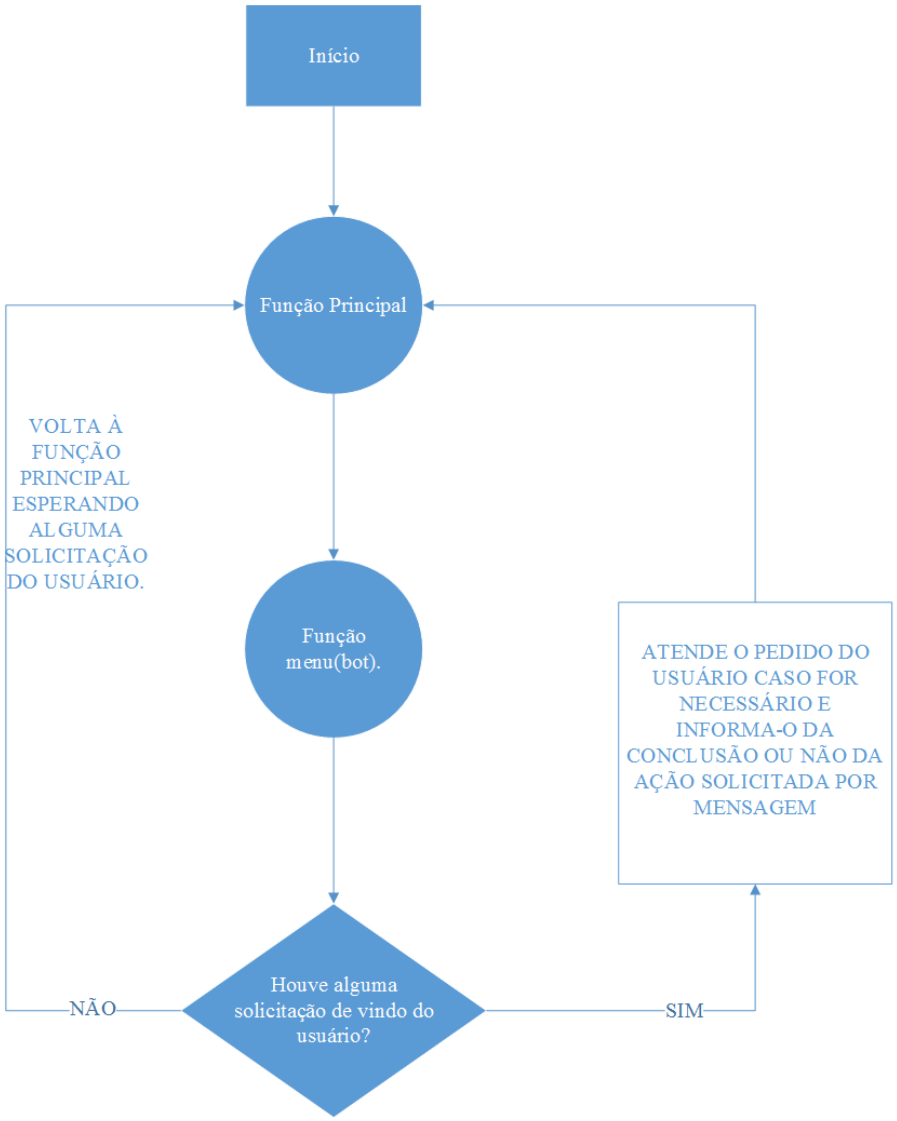

Figura 16 - Lógica para acionamento de equipamentos eletrodomésticos.

Os pedidos do usuário mencionados na Figura 16, estão descritos na Tabela 1, em conjunto com as respectivas ações a serem tomadas e possíveis mensagens enviadas como resposta.

Tabela 1 – comandos para acionamento dos atuadores

Equipamento	Comando Enviado	Ação	Mensagem de Respota
lâmpada	Apagar a lampada ou apagar a lampada desligar a lampada ou desligar a luz ou sem luz ou apagar a luz ou 3	Apaga a lâmpada	Se a lâmpada estiver acesa: “Lâmpada apagada”. Caso contrário: “Ela ja esta apagada, mano”
	acender a lampada ou acender a lampada ou ligar a luz ou acender lampada ou acender a luz ou 2	Acende a lâmpada	Se a lâmpada estiver apagada inicialmente: “Lâmpada acesa”. Caso contrário: “Ela ja está acesa, mano”
	estado da lampada ou 9	envia o estado da lâmpada ao usuário	"A lâmpada está apagada" ou "A lâmpada está acesa"
ventilador	ligar o ventilador ou 4	Liga o ventilador	Se o ventilador estiver anteriormente ao comando desligado, o usuário receberá após o acionamento "Ventilador ligado".Caso contrário: "ele já está ligado, mano"
	desligar o ventilador ou 5	Desliga o ventilador	Se o ventilador estiver anteriormente ao comando ligado, o usuário receberá após o acionamento "Ventilador desligado". Caso contrário: "ele já está desligado, mano"
	estado do ventilador ou 10	envia o estado do ventilador ao usuário	"O ventilador está desligado" ou "O ventilador está ligado"
câmera	modo de seguranca ou foto ou 1	Captura e envia imagem do ambiente ao usuário	image.jpeg

A Figura 17 ilustra o acionamento de uma lâmpada a partir da mensagem enviada via Telegram.

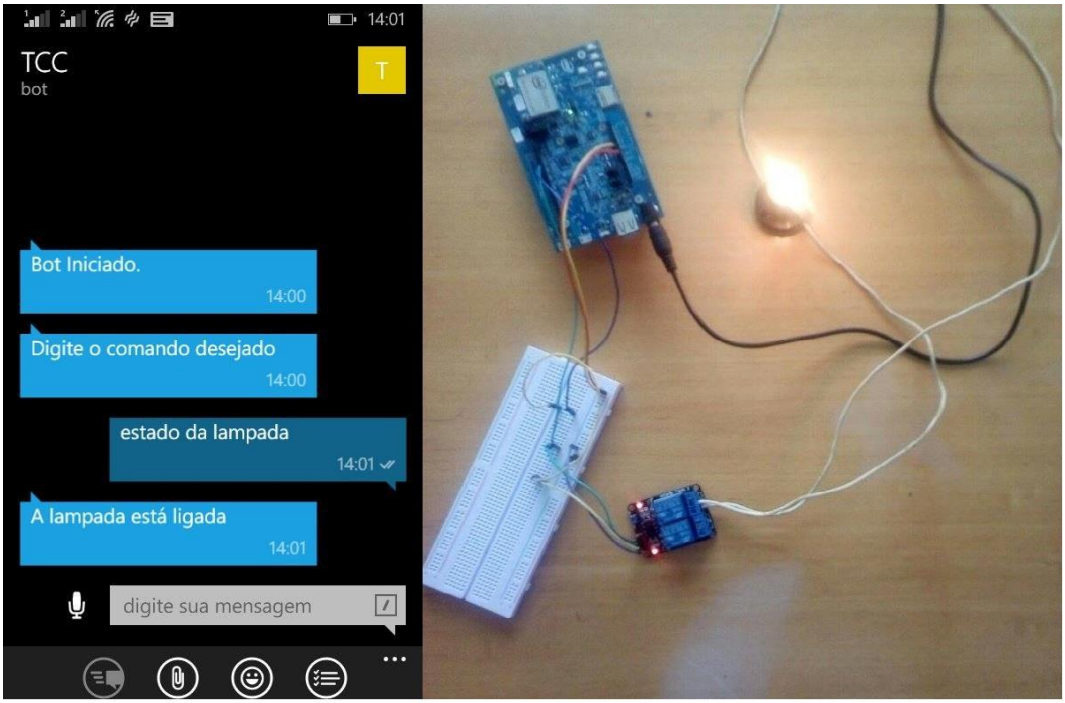

Figura 17 - Circuito ligado (interface e circuito)

5. Operação do equipamento proposto.

Figura 18 - Protótipo completo do projeto proposto

Na Figura 18 tem-se, por fim, o protótipo completo do sistema proposto, com sensores, atuadores, uma lâmpada e um ventilador que simulam a carga que será acionada pelo relé ao comando do usuário e uma webcam para gerar imagens para o sistema de segurança.

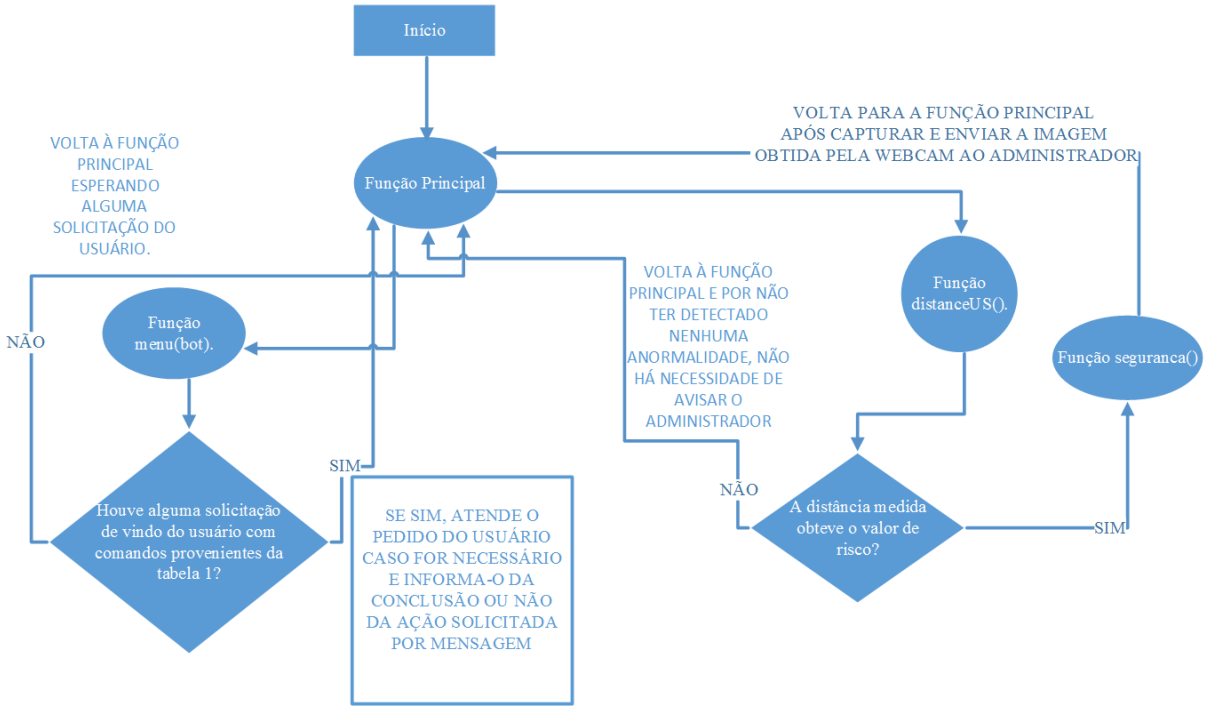

Figura 19 - Fluxograma do sistema

A Figura 19 nos dá uma visão geral da implementação do projeto.

Tem-se um início, onde se importam as bibliotecas e define-se as variáveis. Então, após isso o projeto divide-se em dois braços: Sistema de Segurança e Acionamento de cargas.

No braço de sistema de segurança, mede-se a distância do sensor ao ponto de referência, já conhecida pelo projetista do sistema.

Caso a distância medida fique dentro da margem estipulada como “sem presença” (no caso do projeto, é maior que 100 cm), medirá repetidamente a distância até o sistema acusar “alguma presença” (distância medida menor que 100 cm, por exemplo). Ao se detectar presença, o sistema irá capturar uma imagem usando a webcam e enviar ao usuário cadastrado usando um trocador de mensagem, no caso, o Telegram.

O outro braço, o de acionamentos, irá basicamente esperar o comando do usuário provenientes da tabela 1, ficando preso num loop eterno até houver algum comando de acionamento ou desligamento de carga. Caso ocorra, após atender o pedido do usuário, volta para o loop esperando uma nova ordem.

Observa-se que o projeto possui uma arquitetura ABA (Arquitetura Baseada em Automação), onde os ajustes dos dispositivos são feitos pelos usuários à fim de satisfazer suas necessidades [4]

Na Figura 20, observa-se o acionamento de carga (I) no terminal da Edison ocorrendo após um ciclo de verificação de presença, o que ocorre de meio em meio segundo, não gerando atrasos substancias de resposta.

Na mesma figura, é possível observar em II, tanto no terminal quanto na tela do Telegram, o sistema de segurança sendo ativado.

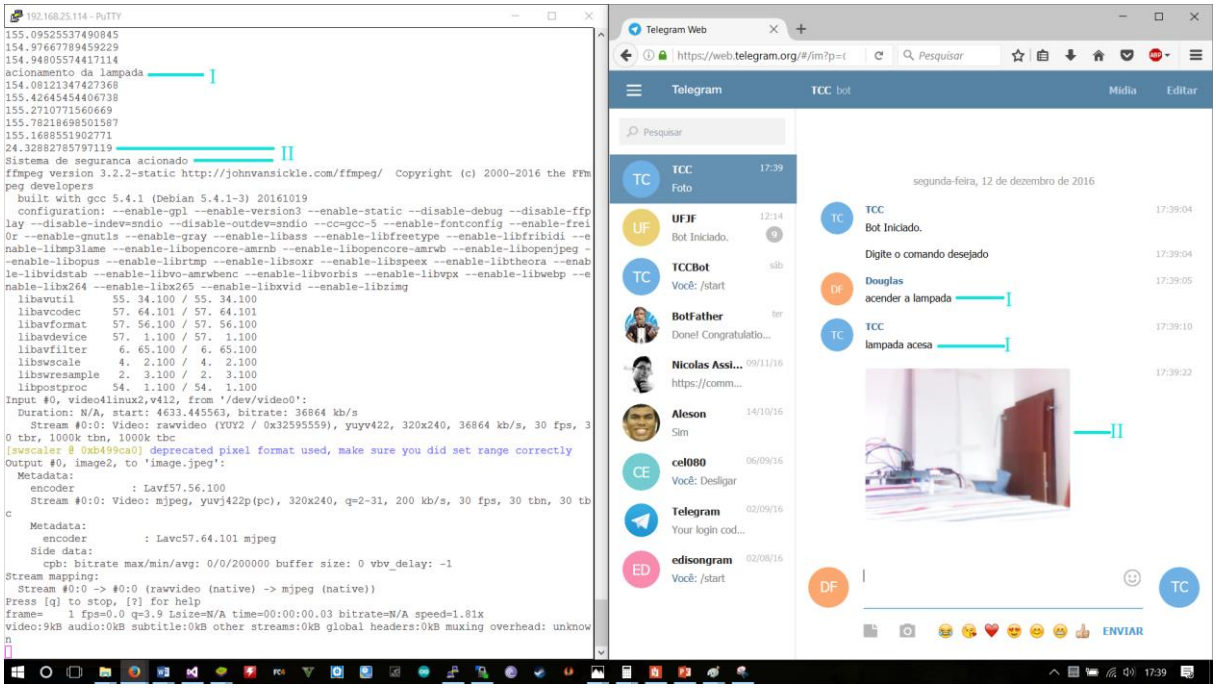

Figura 20 - Funcionamento simultâneo do sistema de segurança com o acionamento de carga.

Na figura 16, observa-se o acionamento de carga (I) no terminal da Edison ocorrendo após um ciclo de verificação de presença, o que ocorre de meio em meio segundo, não gerando atrasos substancias de resposta.

Na mesma figura, é possível observar em II, tanto no terminal quanto na tela do Telegram, o sistema de segurança sendo ativado.

6. CONCLUSÕES

No que foi proposto, o protótipo atendeu com êxito e teve resultados satisfatórios, pois foi possível obter presença em tempo real em algum ambiente, enviando imagens, além de ser possível controlar acionamentos de dispositivos elétricos de até com consumo de até 10A, tudo conversando com a sua casa, mudando assim a concepção de que conhecemos de casa, entrando em uma nova realidade onde inteligência é inserida em equipamentos do nosso dia-a-dia.

Entretanto, como a comunicação é feita via internet, quando a mesma cai, o programa para de rodar. Tentou-se exaustivamente fazer o programa rodar assim que a Edison conseguir-se conectar novamente, mas até o momento dessa apresentação não se conseguiu.

Outro problema é a queda de energia, sendo resolvido sendo alimentado por um nobreak ou até mesmo, projetar uma bateria para alimentar a placa ao cair a energia da rede.

Ou seja, o protótipo atende bem em possibilitar a comunicação com a casa e para o futuro, seria interessante possibilitar ao protótipo ser completamente autônomo, se auto conectando.

Além de resolver os problemas observados em relação à auto recuperação de operação, seria interessante a migração da Arquitetura ABA presente para a ABC.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Introdução à Domótica. Disponível em <http://www.din.uem.br/ia/intelige/domotica/int.htm>
Acessado em 06/08/2016.

[2] Adilson Thomsen, Controlando lâmpadas com Módulo Relé Arduino. Disponível em <http://blog.filipeflop.com/modulos/controle-modulo-rele-arduino.html>
Acessado em 15/12/2016.

[3] O QUE É A DOMÓTICA? Disponível em < http://www.sislite.pt/domus.htm>
Acessado em 15/12/2016.

[4] Lins, Vitor. DOMÓTICA: AUTOMAÇÃO RESIDENCIAL, Recife - PE. Disponível em <http://www.unibratec.edu.br/tecnologus/wp-content/uploads/2010/12/lins_moura.pdf>
Acessado em 15/12/2016

[5] Tonidandel, Flávio. Domótica Inteligente: Automação Residencial baseada em Comportamento, São Bernardo do Campo – SP. Disponível em <http://fei.edu.br/~flaviot/pub_arquivos/WTDIA06.pdf>
Acessado em 15/12/2016

[6]Batista, G.E. Um ambiente de Avaliação de Algoritmos de Aprendizado de Máquina utilizando exemplos. Dissertação (mestrado), Universidade de São Paulo, São Carlos. (1997). Disponível em < http://www.teses.usp.br/teses/disponiveis/55/55134/tde-19082002-234842/pt-br.php>
Acessado em 15/12/2016

[7] Mariotoni C. A. & Andrade E. P. (2002). Descrição de Sistemas de Automação Predial Baseados em Protocolos PLC Utilizados em Edifícios de Pequeno Porte e Residências. CTAI - Revista de Automação e Tecnologia da Informação. Volume 01 nº 1 Janeiro/Junho 2002

[8] Pizzolato, N.D. DOMÓTICA: Aplicabilidade e Sistemas de Automação Residencial. Disponível em <http://essentiaeditora.iff.edu.br/index.php/vertices/article/viewFile/1809-2667.20040015/86>
Acessado em 15/12/2016

[9]SYLVANIA HOME AUTOMATION. Z-Wave Lighting and Appliance Control System Manual. 2005. Disponível em: http://www.unical-usa.com.
Acessado em 15/12/2016

[10] Curvello, André. Criando um bot com Telegram na Intel Edison. Disponível em: <http://www.embarcados.com.br/bot-com-telegram-na-intel-edison/ >
Acesso em 21 de junho de 2016.

[11] Thomsen, Adilson. Como Conectar o Sensor Ultrassônico HC-SR04 ao Arduino. Disponível em :http://blog.filipeflop.com/sensores/sensor-ultrassonico-hc-sr04-ao-arduino.html

[12] Telegram Bot Platform. Disponível em: https://telegram.org/blog/bot-revolution
Acessado em 15/12/2016

[13] Disponível em: < https://gist.github.com/gaiar/3dc7a39261927d5e20e3>
Acessado em 15/12/2016

[14] Disponível em <https://pypi.python.org/pypi/python-telegram-bot>
Acessado em 15/12/2016

APÊNDICE 1 – Instalação do Python 3.4.3 e as bibliotecas necessárias

- Baixar, compilar e instalar o Python3 [13]

>>wget https://www.python.org/ftp/python/3.5.0/Python-3.5.0.tgz

>>tar xvf Python-3.5.0.tgz

>>cd Python-3.5.0

>>./configure --prefix=/usr --enable-shared

>>make -j2

>>make install

- Baixar, compilar e instalar o libmraa [13]

>> git clone https://github.com/intel-iot-devkit/mraa.git

>>cd mraa

>> mkdir build

>> cd build

>> cmake -DCMAKE_INSTALL_PREFIX:PATH=/usr -DBUILDPYTHON3=ON -DPYTHON_INCLUDE_DIR=/usr/include/python3.5m/ -DPYTHON_LIBRARY=/usr/lib/libpython3.so ..

>>make -j2

>>make install

- Baixar, compilar e instalar o libmraa [13]

>> git clone https://github.com/intel-iot-devkit/upm.git

>> cd upm

>> mkdir build

>> cd build

>>cmake -DPYTHON_INCLUDE_DIR=/usr/include/python3.5m/ -DPYTHON_LIBRARY=/usr/lib/libpython3.so -DCMAKE_INSTALL_PREFIX:PATH=/usr ..

>>make -j2

>>make install

- Instalar ou atualixar a biblioteca do telegrama [14]

```
>> pip install python-telegram-bot --upgrade
```

APÊNDICE 2 – Scripts usados.

- Script em Python para obter o id de usuário

```
1. import telegram
2. # e tambem importamos a biblioteca sleep de time, para poder criar
3. # "delay" no codigo Python.
4. from time import sleep
5.
6. # Faz os imports de bibliotecas para tratar erros de conexao URL.
7. try:
8.     from urllib.error import URLError
9. except ImportError:
10.     from urllib2 import URLError
11.
12. # Criamos uma rotina main para gerir o codigo principal
13. def main():
14.         # Variavel update_id - usada pelo Telegram
15.         update_id = None
16.
17.         # Criamos um objeto bot inserindo o Token fornecido
18.         # pelo The BotFather
19.
20.         bot = telegram.Bot('326945283:AAG3As-Dj4b8rkJRS6U2I4GezJFjGtpzkvw')
21.
22.
23.         print ('Bot Telegram iniciado...')
24.
25.         # Loop infinito - programa em execucao
26.         while True:
27.             try:
28.                 update_id = edisonGramBot(bot, update_id)
29.                 print("teste")
30.             except telegram.TelegramError as e:
31.                 # Se ocorrer algum problema, lentidao, por ex:
32.                 if e.message in ("Bad Gateway", "Timed out"):
33.                     sleep(100) # Espera 1 segundo...
34.                 else: # Caso contrario, lanca excessao.
```

```
                    raise e
            except URLError as e:
                # Ha problemas de rede na execucao...
                sleep(1)


def edisonGramBot(bot, update_id):
    # Requisita atualizacoes depois da ultima id de update - update_id

    # bot.getUpdates(offset, timeout) - offset eh o ponto de partida em
    # que comeca a procurar novas atualizacoes de mensagens, timeout eh
    # tempo minimo de espera para retorno da requisicao de resposta.
    for update in bot.getUpdates(offset=update_id, timeout=10):

        # o chat_id eh a id do chat de comunicacao Telegram
        # eh necessaria para o bot identificar a conversa e
 # gerar e enviar a resposta
        chat_id = update.message.chat_id
        #vai printar o chat_id de quem mandou msg.
        id = chat_id
        aux = repr(id)
        aux2="o seu id eh : "+ aux
        print(chat_id)


        # atualiza o indice update_id - para ref novas mensagens
        update_id = update.update_id + 1
        #imprime o numero da msg enviada.
        print(update_id)

        # Captura a mensagem de texto enviada ao bot no dado chat_id
        message = update.message.text

        if message:
            # Envia a mensagem para o chat_id especifico, com a mensagem
     # parametrizada.
            bot.sendMessage(chat_id=chat_id, text=aux2)

    # retorna o ultimo update_id para servir de referencia
    return update_id

# Rotina essencial para executar a rotina main() quando o codigo python
# eh executado.
if __name__ == '__main__':
    main()
```

- Script usado para controlar a casa.

```
#Importar as bibliotecas
import telegram
from telegram.error import NetworkError, Unauthorized
from time import sleep
import random
import telegram
from telegram.error import NetworkError, Unauthorized
from time import sleep
```

```
9.  import random
10. import string
11. import os
12. import mraa
13. import time
14. import codecs
15.
16.
17.
18. i=1 #flag
19. k=1 #contador teste
20. update_id = None # limpa a variável que armazena a quantidade de mensagem
21. CHAT_id=258541615  # id do usuário que irá operar o sistema
22. CHAT_id2=0 #id de um segundo usuário ainda nao definido
23. administrador = CHAT_id #definição do administrador do sistema
24.
25.
26. ventilador = mraa.Gpio(8) #atribuição do I/O da lampada
27. ventilador.dir(mraa.DIR_OUT) #configuração do I/O como saída
28. ventilador.write(1) #coloca o I/O em nível lógico alto
29.
30. lampada = mraa.Gpio(7)  #atribuição do I/O da lampada
31. lampada.dir(mraa.DIR_OUT) #configuração do I/O como saída
32. lampada.write(1) #coloca o I/O em nível lógico alto
33.
34.
35. trig = mraa.Gpio(3)  #atribuição do I/O do trig do sensor HC-SR04
36. echo = mraa.Gpio(5)  #atribuição do I/O do echo do sensor HC-SR04
37.
38. # definindo o bot usado pelo token
39. bot = telegram.Bot('326945283:AAG3As-Dj4b8rkJRS6U2I4GezJFjGtpzkvw')
40.
41. #envio de mensagem ao usuário que o sistema foi iniciado
42. bot.sendMessage(chat_id=CHAT_id, text="Bot Iniciado.")
43. #envio de mensagem ao usuário dizendo que o sistema está pronto para receber algum
    comando.
44. bot.sendMessage(chat_id=CHAT_id, text='Digite o comando desejado')
45.
46.
47. #Cria-se uma rotina main para gerir a função principal
48. def main():
49.     # Variavel update_id - usada pelo Telegram
50.     try:
51.         update_id = bot.getUpdates()[0].update_id
52.     except IndexError:
53.         update_id = None
54.         #loop que, enquanto for verdade, irá chamar a função distanceUS(),
    menu(bot) e, caso entrar no parâmetro do if (distance<100), chama a função
    seguranca
55.     while True:
56.         try:
57.             t1 = time.time()
58.             distanceUS()
59.             print(distance)
60.             if distance < 100:
61.                 if (CHAT_id2 == 0):
62.                     print("Sistema de seguranca acionado")
63.                     seguranca()
64.
65.             menu(bot)
```

```
66.         except NetworkError:
67.
68.             sleep(10000000000)
69.
70.     exit()
71.     #função menu(bot), responsável pela operação de acionamentos e desligamentos
    de carga e atuadores
72. def menu(bot):
73.     global update_id
74.     global i
75.     global k
76.
77.     global CHAT_id2
78.     global conta_trouxa
79.     global CHAT_id
80.
81.     for update in bot.getUpdates(offset=update_id, timeout=1):
82.
83.         chat_id = update.message.chat_id
84.         update_id = update.update_id + 1
85.         message = update.message.text
86.
87.         if (chat_id==CHAT_id or chat_id==CHAT_id2):
88.
89.             if message:
90.                 if(message.lower()=='1' or message.lower()=='modo de seguranca' or
    message.lower()=='foto'): #se a mensagem for alguma dessas
91.                     os.system ("./ffmpeg -s 320x240 -f video4linux2 -i /dev/video0
    -vframes 1 image.jpeg") #captura uma imagem
92.                     time.sleep(1) #espera um segundo
93.                     bot.sendPhoto(chat_id=administrador, photo=open('image.jpeg',
    'rb')) #envia a imagem ao administrador
94.                     os.system ("rm image.jpeg") #apaga para otimizar a memória de
    armazenamento do sistema embarcado
95.                     #se alguma das mensagens enviadas for as de baixo
96.                 elif(message.lower()=='acender a lampada' or
    message.lower()=='acender a lampada' or message.lower()=='ligar a luz' or
    message.lower()=='2' or message.lower()=='lampada' or message.lower()=='acender
    lampada' or message.lower()=='acender a luz' or message.lower()=='lampada' ):
97.                     #verifica-se o estado da lampada. Se estiver apagada
98.                     if (lampada.read()==1):
99.                         lampada.write(0) #acende-a
100.                            bot.sendMessage(chat_id=CHAT_id, text="lampada acesa") #
    e avisa ao usuário
101.                            print("acionamento da lampada") #mensagem no terminal
102.                    else: #se não
103.
104.                            bot.sendMessage(chat_id=CHAT_id, text="Ela ja esta
    acesa, mano") #avisa que ela já está acesa
105.
106.                #se alguma das mensagens enviadas for as de baixo
107.                elif(message.lower()=='Apagar a lampada' or
    message.lower()=='apagar a lampada' or message.lower()=='desligar a lampada' or
    message.lower()=='3' or message.lower()=='desligar a luz' or message.lower()=='sem
    luz'  or message.lower()=='apagar a luz' or message.lower()=='sem luz'):
108.                    #verifica-se o estado da lampada. Se estiver apagada
109.                    if (lampada.read()==0):
110.                        lampada.write(1) #acende-a
111.                        print("desligamento da lampada") #printa no terminal
```

```
112.                            bot.sendMessage(chat_id=CHAT_id, text="lampada apagada")
    #avisa ao usuário
113.                        else:
114.                          bot.sendMessage(chat_id=CHAT_id, text="Ela ja esta
    apagada, mano")  #avisa que a lâmpada já estava apagada
115.                           print("desligamento da lampada")
116.
117.                           ###para o ventilador é o mesmo sistema da lampada, só
    verificar o tipo de ligação do relé, se está normalmente aberto ou normalmente
    fechado
118.                    elif(message.lower()=='ligar o ventilador'  or
    message.lower()=='3' or message.lower()=='Ligar o ventilador'):
119.                        if (ventilador.read()==1):
120.                            ventilador.write(0)
121.                            bot.sendMessage(chat_id=CHAT_id,text="Ventilador
    Ligado")
122.                        else:
123.                          bot.sendMessage(chat_id=CHAT_id, text="Ele ja esta
    desligado, mano")
124.
125.                    elif(message.lower()=='desligar o ventilador' or
    message.lower()=='4' or message.lower()=='desligar o ventilador'):
126.                        if (ventilador.read()==0):
127.                            ventilador.write(1)
128.                            bot.sendMessage(chat_id=CHAT_id,text="Ventilador
    desligado")
129.                        else:
130.                            bot.sendMessage(chat_id=CHAT_id, text="Ele ja esta
    desligado, mano")
131.
132.                            #mensagem para o usuário solicitar o estado da carga,
    que em questão, é a lâmpada
133.                    elif(message.lower()=='estado da lampada' or
    message.lower()=='9'):
134.                        if ((lampada.read()==0)):
135.
136.                            aux2="A lampada está ligada"
137.                            bot.sendMessage(chat_id=chat_id, text=aux2)
138.                        else:
139.                            bot.sendMessage(chat_id=chat_id, text='A lâmpada está
    apagada')
140.
141.                    elif(message.lower()=='estado do ventilador' or
    message.lower()=='10'):
142.                        if ((ventilador.read()==1)):
143.
144.                            aux2="O ventilador está: ligado"
145.                            bot.sendMessage(chat_id=chat_id, text=aux2)
146.                        else:
147.                            bot.sendMessage(chat_id=chat_id, text='O Ventilador está
    desligado')
148.
149.  def distanceUS():
150.      tZero = time.time()
151.
152.      trig.write(0)
153.      time.sleep(0.04)
154.
155.      trig.write(1)
156.      time.sleep(0.1)
```

```
157.
158.
159.        sig = None
160.        nosig = None
161.        etUS = None
162.        global distance
163.
164.        while echo.read() == 0:
165.                trig.write(0)
166.                sig = None
167.                nosig = None
168.                etUS = None
169.                nosig = time.time()
170.                #print("Estagio 1")
171.
172.        while echo.read() == 1:
173.                sig = time.time()
174.                #print("Estagio 2")
175.        if sig == None or nosig == None:
176.            return 0
177.
178.        # et = Elapsed Time
179.        etUS = sig - nosig
180.
181.        distance =  etUS * 17150
182.        #print( distance )
183.        return distance
184.
185.
186.    def seguranca():
187.        global update_id
188.        global i
189.        #global temp
190.        #global j
191.        global CHAT_id2
192.        global conta_trouxa
193.        global CHAT_id
194.      #  if
195.        os.system ("./ffmpeg -s 320x240 -f video4linux2 -i /dev/video0 -vframes 1
    image.jpeg")
196.        time.sleep(1)
197.        bot.sendPhoto(chat_id=administrador, photo=open('image.jpeg', 'rb'))
198.        os.system ("rm image.jpeg")
199.        time.sleep(1)
200.
201.
202.    if __name__ == '__main__':
203.        main()
```