UNIVERSIDADE DO OESTE DE SANTA CATARINA
UNOESC – UNIDADE CHAPECÓ

RODRIGO BATISTELLO

AUTOMAÇÃO RESIDENCIAL UTILIZANDO *RASPBERRY PI* E ANDROID

CHAPECÓ, SC
2014

RODRIGO BATISTELLO

AUTOMAÇÃO RESIDENCIAL UTILIZANDO *RASPBERRY PI* E ANDROID

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação, Área das Ciências Exatas e da Terra, da Universidade do Oeste de Santa Catarina – UNOESC Unidade de Chapecó como requisito parcial à obtenção do grau de bacharel de Sistemas de Informação.

Orientador: Prof. Jose Luiz Cunha Quevedo

Chapecó, SC
2014

RODRIGO BATISTELLO

AUTOMAÇÃO RESIDENCIAL UTILIZANDO *RASPBERRY PI* E ANDROID

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação, Área das Ciências Exatas e da Terra, da Universidade do Oeste de Santa Catarina – UNOESC Unidade de Chapecó como requisito parcial à obtenção do grau de bacharel de Sistemas de Informação.

Aprovado em: ____/____/_____

BANCA EXAMINADORA

_____
Prof. Jose Luiz Cunha Quevedo
Universidade do Oeste de Santa Catarina
Nota atribuída:

_____
Prof. Cristiano Agosti
Universidade do Oeste de Santa Catarina
Nota atribuída:

_____
Prof. Moacir Alves
Universidade do Oeste de Santa Catarina
Nota atribuída:

A minha família pelo apoio irrestrito em todos os momentos de minha vida.

AGRADECIMENTOS

Agradeço a Deus por tudo, principalmente pela saúde e a força para concluir este trabalho.

A minha família, meu pai Ary, minha mãe Lorena e minhas irmãs Francieli e Eveline pelo apoio e o incentivo em todos os momentos difíceis.

Ao meu orientador Jose Luiz por ter aceitado fazer parte deste desafio e por ter me ajudado a concluí-lo.

Aos meus amigos e colegas de curso, Bruna, Franciele e Rafael por terem me ajudado na resolução de problemas e no esclarecimento de dúvidas.

E por fim, a todos os professores e outras pessoas que de alguma forma contribuíram para a realização deste trabalho.

“A persistência é o menor caminho do êxito”.
Charles Chaplin

RESUMO

A automação residencial surgiu entre as décadas de 1970 e 1980 nos Estados Unidos, com o intuído de facilitar a vida das pessoas, permitindo um controle centralizado e integrado de diferentes tipos de equipamentos de uma residência. O custo de instalação desse tipo de tecnologia geralmente é elevado e feito apenas por profissionais capacitados, isso motivou o desenvolvimento de um protótipo de automação residencial utilizando tecnologias mais populares, no caso o *Raspberry Pi*, uma placa de expansão e um dispositivo móvel com plataforma Android. Por meio de uma pesquisa exploratória, cujas técnicas utilizadas foram pesquisas documentais e bibliográficas a fim de compreender e explicar do que se trata a automação residencial e as tecnologias utilizadas, desenvolveu-se um protótipo que possibilita aos moradores de uma residência controlarem remotamente a iluminação, enviar comandos para acionamento de equipamentos eletrônicos, controlar alarme e som ambiente, permitir a visualização da temperatura e humidade bem como o monitoramento de uma câmera de segurança. Também foi desenvolvido um controle de agendamento para acionar os equipamentos em determinados dias e horários. Isso tudo resultou em um sistema de automação perfeitamente utilizável. As tecnologias abordadas adaptaram-se com mérito ao projeto, apontando assim que o uso do *Raspberry Pi* em sistemas de automação é perfeitamente viável bem como a integração com dispositivos móveis que utilizam a plataforma Android.

Palavras-chave: Automação residencial, *Raspberry Pi,* Android.

ABSTRACT

Residential automation arose between the decades of 1970 and 1980 in the United States, with the aim of facilitating people's lives, allowing a centralized and integrated control of different types of equipment of a residence. The cost of installation of this type of technology is usually high and done only by trained professionals, that motivated the development of a prototype of residential automation using most popular technologies, in case the Raspberry Pi, an expansion card and a mobile device with Android platform. Through an exploratory research, whose techniques used were documentary and bibliographical researches in order to understand and explain what home automation is all about and the technologies used, it was developed a prototype that allows residents of a residence to control remotely the lighting, to send commands to start electronic equipment, to control the alarm and the sound of the environment, to allow the display of the temperature and humidity as well as the monitoring of a security camera. It was also developed a scheduling control to trigger the equipment on certain days and times. This all resulted in a perfectly usable automation system. The technologies addressed have adapted with merit to the project, pointing out that the use of Raspberry Pi in automation systems is perfectly feasible as well as the integration with mobile devices using the Android platform.

Keywords: Home automation, Raspberry Pi, Android.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES E SIGLAS

ADT	*Android Developer Tools*
ARM	*Advanced RISC Machine*
FTP	*File Transfer Protocol*
GB	*Gigabyte*
Gbps	*Gigabit* por segundo
GPIO	*General Purpose Input/Output*
GPL	*General Public License*
HDMI	*High-Definition Multimedia Interface*
IDE	*Integrated Development Environment*
IEEE	*Institute of Electrical and Electronics Engineers*
LAN	*Local Area Networks*
MB	*Megabyte*
Mbps	*Megabit* por segundo
MDF	*Medium Density Fiberboard*
MER	Modelo Entidade-Relacionamento
MHz	*Mega Hertz*
OHA	*Open Handset Alliance*
PDA	*Personal Digital Assistant*
RAM	*Random Access Memory*
SD	*Secure Digital*
SSH	*Secure Shell*
USB	*Universal Serial Bus*
UTF-8	*8-bit Unicode Transformation Format*
VDI	Voz, Dados e Imagem
WLAN	*Wireless Local Area Netwok*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A automação residencial ou domótica como também é conhecida, da união do latim "*Domus"* que significa casa e "robótica" controle automatizado de algo, veio para permitir aos usuários um controle automatizado e centralizado das residências, agregando mais facilidade e qualidade de vida as pessoas.

Com a popularização dos *smartphones* e *tablets* surgiu a possibilidade de integrar as tecnologias móveis com a automação, a fim de proporcionar aos usuários controle sobre suas residências mesmo à distância proporcionando uma nova experiência de vida aos moradores dessas "casas inteligentes".

Geralmente o custo para implantação desse tipo de sistema é bastante elevado e feito apenas por profissionais capacitados, o que torna a mesma inacessível para grande parte da população. No entanto o surgimento de novas tecnologias proporcionou a construção de sistemas de automação bastante similares aos já comercializados e com um custo relativamente menor.

É por esses fatores que este trabalho justifica-se na construção de um sistema de automação residencial composto por um dos menores computadores do mundo, denominado *Raspberry Pi,* juntamente com uma placa de expansão própria para esta finalidade e um *smartphone* com sistema Android. Neste sistema será possível controlar iluminação, alarme, som ambiente, visualizar câmera de segurança, temperatura e humidade, criar agendamentos, entre outros.

Tudo isso no intuito de facilitar ainda mais a vida das pessoas, permitir um controle centralizado e sem a necessidade de estar presente no local, aumentar a segurança da residência, a qualidade de vida dos moradores, além de comprovar que essas novas tecnologias se adaptam perfeitamente a esta finalidade.

1.1 OBJETIVOS

A seguir apresenta-se o objetivo geral deste trabalho, bem como os objetivos específicos.

1.1.1 Objetivo geral

Desenvolver um protótipo de automação residencial onde o usuário poderá controlar remotamente sua residência através de um dispositivo móvel com plataforma Android, enviando comandos para controlar iluminação, equipamentos eletrônicos, sistema de alarme, entre outros.

1.1.2 Objetivos específicos

Para atingir o objetivo geral, consideram-se os seguintes objetivos específicos:

- Estudar e compreender o funcionamento de uma residência automatizada, levantando as principais aplicações e vantagens.
- Demonstrar a aplicação de novas tecnologias em soluções de automação residencial.
- Construir um protótipo para representar a aplicação da automação residencial.
- Construir as ligações elétricas entre o *Raspberry Pi*, placa de automação e o protótipo, possibilitando o acionamento dos equipamentos e sensores.
- Desenvolver um aplicativo para Android que possibilite o controle da automação residencial a distância, enviando comandos para ligar e desligar a iluminação, eletroeletrônicos, alarme, entre outros.
- Desenvolver um aplicativo servidor que ficará rodando no *Raspberry Pi*, recebendo e executando os comandos encaminhados pelo dispositivo móvel.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado da seguinte maneira: o capítulo 2 ou revisão da literatura, é o local onde será apresentado todo referencial teórico necessário para elaboração do trabalho, tratando as principais características e vantagens da

automação residencial e abordando as tecnologias utilizas, como: *Raspberry Pi* e Android, além das linguagens de programação Java e Python.

Na sequência serão apresentados os capítulos 3 e 4 que contém informações sobre o campo e a área do estudo além das metodologias utilizadas e questões de pesquisa que instigaram a elaboração do trabalho.

O capítulo 5 apresenta os resultados deste trabalho e tudo o que foi feito para atingir os objetivos e responder as questões de pesquisa. Ele descreve todos os passos executados, como a instalação do sistema operacional no *Raspberry Pi,* instalação de pacotes de aplicativos no sistema, criação do protótipo, desenvolvimento e funcionamento do sistema proposto junto ao protótipo criado.

Por último temos o capítulo 6, onde está a conclusão deste trabalho, ou seja, tudo que foi possível aprender e concluir com a realização do mesmo, bem como ideias para trabalhos futuros.

2 REVISÃO DA LITERATURA

A revisão da literatura desse trabalho aborda uma visão geral e atualizada dos assuntos tratados, com enfoque principal na automação residencial ou domótica. Também serão apresentadas as tecnologias utilizadas no seu desenvolvimento como *Raspberry Pi,* Android, Java e Python.

2.1 AUTOMAÇÃO RESIDENCIAL

Segundo Muratori e Dal Bó (2011, p. 1) 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".

A automação residencial também é conhecida como domótica (do latim "*Domus"* que significa casa e "robótica" controle automatizado de algo), casa inteligente, casa automática ou *retrofitting* (ato de introduzir uma modificação em algo já construído). Alguns autores costumam achar mais adequado o termo domótica, por ser mais abrangente e largamente utilizado na Europa, porém no Brasil é mais comum o uso da tradução literal do termo *home automation,* denominação americana mais restrita, ou automação residencial. (MURATORI; DAL BÓ, 2011; BOLZANI, 2004).

Sendo assim a automação residencial pretende identificar todas as tecnologias que permitem tornar automática uma série de operações e tarefas dentro de uma residência (PRUDENTE, 2011).

Para Prudente (2011) a domótica pode ser aplicada em todas as atividades dentro de uma habitação, integrando e conectando diferentes tipos de equipamento e instalações, alguns dos exemplos que ele cita são: controle de iluminação e dimerização (luminosidade), sistema de ar condicionado, ligar e desligar televisões, comandos de veneziana, porta, portão eletrônico, controle de parâmetros ambientais e atmosféricos, comando e controle para cada tipo de eletrodoméstico, sistema de alarme antifurto, controle de acesso, detecção de incêndio, vazamento de gás, perda

de água, câmeras de vigilância, entretenimento, *home theater,* internet, tele socorro e outros auxílios para idosos e deficientes físicos.

A automação residencial é baseada na automação industrial, muito mais conhecida e difundida. Devido a diferença entre os dois tipos de ambientes, projetos específicos para cada área têm sido desenvolvidos. A automação em projetos de pequeno e médio porte com características comerciais e residenciais além dos primeiros módulos inteligentes começaram a surgir entre as décadas de 1970 e 1980. Nos Estados Unidos, no fim dos anos 70 surgia os primeiros módulos inteligentes, com comandos enviados pela própria rede elétrica, tratavam-se de soluções simples e não integradas como ligar remotamente algum equipamento ou luzes. Já nos anos 80 algumas companhias começaram a desenvolver sistemas de automação, e em 1996 havia mais de 4 milhões de edifícios e residências automatizadas (BOLZANI, 2004; MURATORI; DAL BÓ, 2011).

Hoje existem no mercado empresas que oferecem um conjunto completo de recursos, onde o usuário pode escolher entre as opções disponíveis, além do mais os dispositivos podem ser acionados e controlados pela mesma interface, seja remoto, por voz ou telefone. Esses sistemas podem também ser programados para simular presença e assim expulsar possíveis intrusos, ou até mesmo chamar a polícia. (BOLZANI, 2004).

Segundo Bolzani (2004, p. 60) as características de um bom sistema inteligente são:

- Capacidade de integrar todos os sistemas;
- Atuação em condições variadas;
- Memória;
- Noção temporal;
- Fácil relação com o usuário;
- Facilidade de programação;
- Autocorreção.

Para Muratori e Dal Bó (2011) a automação residencial pode ser dividida em 4 classes, cada uma delas compreende as seguintes opções:

- Automação da instalação elétrica: compreende o fornecimento de controladores e atuadores: *softwares* de controle e supervisão, relés, interface, temporizadores e sensores;

- Controles remoto universais: fornecimento de controles fixos em paredes, painéis móveis, licenças de *software* de integração para Iphone, celulares, entre outros;
- Gestão de energia: sistema de gerenciamento do consumo energético, medição do consumo de água, equipamentos para proteção elétrica e de geração de fontes de energia alternativas;
- Acessórios e complementos: compreende motorização de persianas, toldos e cortinas, pisos aquecidos, aspiração central a vácuo, desembaçadores de espelhos, irrigação automática, fechaduras elétricas, equipamentos de controle de acesso, media centers, ativos de rede e telefonia.

A aplicação de sistemas inteligentes e automatizados trazem inúmeros benefícios aos seus utilizadores.

> Automatizando os sistemas, consegue-se um aproveitamento melhor da luminosidade ambiente, controlando luzes e persianas e mantendo sempre a temperatura ideal, mas sem desperdício, obtendo-se uma redução no consumo de energia. Um ambiente inteligente é aquele que otimiza certas funções inerentes à operação e administração de uma residência. É como se ela tivesse vida própria, com cérebro e sentidos (BOLZANI, 2004, p. 60).

Entre as vantagens obtidas com a aplicação de uma automação residencial Prudente (2011, p. 3) destaca:

- Maior conforto: deixam o ambiente mais acolhedor e agradável, permitem gerir e controlar parâmetros que incidem diretamente sobre a qualidade de vida das pessoas;
- Maior segurança: tanto no que diz respeito as pessoas, quanto eventos perigosos. É possível se precaver contra furtos, ou no caso de um sinistro como incêndio, vazamentos de gás, entre outros;
- Maior versatilidade: possibilita variar a instalação e as funções dos componentes com o uso de *softwares* dedicados;
- Maior economia na gestão da instalação: é possível obter-se um controle total sobre a energia, iluminação, aquecimento, sistema de ar condicionado, entre outros.

Além das vantagens citadas acima a aplicação de um sistema de automação residencial pode trazer benefícios também às pessoas com dificuldades de locomoção ou com algum tipo de deficiência física. Existem aparelhos específicos que dão uma certa liberdade à elas, porém costumam ser caros e delicados. Com os sistemas

residenciais inteligentes, muitas alternativas mais baratas e flexíveis têm surgido. Para essas pessoas faz muita diferença poder ligar e desligar luzes e equipamentos de uma cadeira de rodas ou da cama (BOLZANI, 2004).

Em economias mais desenvolvidas o cenário é bastante propício para as chamadas “casas inteligentes”, algumas contribuições importantes para isso foram a popularização de diversas tecnologias e à queda nos preços. As ofertas abundantes de serviços de comunicação como a internet também ajudaram. (MURATORI; DAL BÓ, 2011).

Pesquisas realizadas nos Estados Unidos revelaram que 84% dos construtores de residências entendem que incorporar tecnologia aos empreendimentos é um importante diferencial de mercado, consumidores mais novos adquirindo seu primeiro imóvel, são mais exigentes nesse quesito e o uso de sistemas automatizados com apelo a sustentabilidade são cada vez mais requisitados. Entre as tecnologias emergentes que tem possibilidade de crescimento nos próximos anos destacam-se media centers, monitoramento a distância, controle de iluminação e o *home care* (modalidade de prestação de serviços na área da saúde visando à continuidade do tratamento hospitalar em domicílio) (MURATORI; DAL BÓ, 2011).

O crescimento da utilização dessas tecnologias e a previsão para os próximos anos pode ser observado na tabela 1.

Tabela 1 – Utilização de tecnologias relacionadas a automação residencial

Tecnologia	**2003**	**2004**	**2005**	**2006**	**2015 (Previsão)**
Cabeamento estruturado	42%	61%	49%	53%	80%
Monitoramento de segurança	18%	28%	29%	32%	81%
Multiroom áudio	9%	12%	15%	16%	86%
Home Theater	9%	8%	11%	12%	86%
Controle de iluminação	1%	2%	6%	8%	75%
Automação integrada	0%	2%	6%	6%	70%
Gerenciamento de energia	1%	5%	11%	11%	62%

Fonte: Adaptado de MURATORI; DAL BÓ (2011, p.3).

Apesar de muitas pessoas ainda pensarem nessas casas com certa visão futurista, o potencial de crescimento desse mercado é enorme, tanto que fez surgir novas profissões, como a de “integrador de sistemas”, pessoa responsável pela especificação, fornecimento, instalação, programação e pós-venda de sistemas de automação residencial (PRUDENTE, 2011; MURATORI; DAL BÓ, 2011).

Aqui no Brasil o panorama tende à mudar em um pequeno espaço de tempo, à competição do mercado imobiliário cada vez mais acirrada e a utilização de diferenciais tecnológicos podem fazer da automação um fator decisivo para atingir consumidores com necessidades especificas, como segurança, entretenimento, acessibilidade, trabalho em casa, conforto, conveniência e economia de energia, aproximando cada vez mais o mercado doméstico brasileiro dos padrões internacionais. (MURATORI; DAL BÓ, 2011).

Existem diversas tecnologias que podem ser utilizadas no desenvolvimento de sistemas de automação. Neste trabalho foi optado pelo uso do *Raspberry Pi* como controlador principal do sistema, entenderemos um pouco mais sobre esta tecnologia na sequência.

2.2 RASPBERRY PI

Os computadores já fazem parte de nossas vidas e são indispensáveis, tanto no uso pessoal quanto profissional. Eles nos auxiliam nas tarefas cotidianas, na comunicação e na automatização de processos.

O *Raspberry Pi* nada mais é que um dos menores computadores do mundo, suas dimensões são próximas a de um cartão de crédito e seu sistema operacional padrão é o Linux. Ele pode ser usado para realizar boa parte das tarefas de um computador comum e também nos mais diferentes tipos de projetos (NETTO, 2013).

A notícia do desenvolvimento do *Raspberry Pi* veio à tona em maio de 2011 e foi muito bem recebida. Os próprios idealizadores ficaram surpresos, isso porque ele despertou o interesse de programadores, profissionais da área de automação e entusiastas em geral, mesmo antes do início de sua produção (NETTO, 2013, p. 13).

De acordo com Netto (2013) e a Raspberry Pi Foundation ([2013?]) o projeto de desenvolvimento teve início em 2006, quando um professor da universidade de Cambridge, chamado Eben Upton, notou que o conhecimento sobre computadores dos novos alunos de ciências da computação tinha diminuído em relação aos alunos da década de 1990, isso porque antigamente para se usar um computador tinha que saber um pouco de programação, ele imaginou então que se existisse um computador flexível o bastante para ser uma ferramenta de aprendizado, com baixa potência e

custo de fabricação que pudesse ser dado aos ingressantes poderia ajudar muito. Porém depois de um tempo eles perceberam o verdadeiro potencial do projeto, criaram uma instituição de caridade chamada Raspberry Pi Foundation e o *Raspberry Pi* deixou de ser destinado apenas aos acadêmicos e passou a ser uma verdadeira ferramenta educacional, que poderia ser utilizada por crianças de todo o mundo.

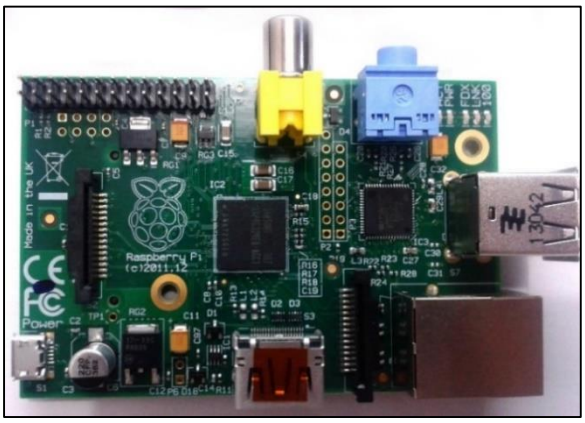
No entanto, quanto mais a data de lançamento se aproximava mais era o interesse das pessoas pelo pequeno computador, então percebeu-se que ele não serviria mais apenas para ensinar crianças, mas também como uma plataforma alternativa para profissionais experientes de todo o mundo (NETTO, 2013).

Inicialmente o computador não se chamaria *Raspberry Pi*, Eben Upton pensou em batiza-lo de “ABC Micro”, em homenagem aos computadores dos anos 90 que eram utilizados para ensinar programação às crianças na escola, esses computadores eram chamados de “BBC Micros”. Porém ele foi recebendo ideias de outros colaboradores até que o nome foi mudado. O termo “*Raspberry*” que em português significa framboesa é uma alusão à tradição de batizar computadores com o nome de fruta, enquanto “*Pi*” é uma referência ao Python, que é a linguagem de programação nativa do *Raspberry Pi* (NETTO, 2013; RASPBERRY PI FOUNDATION, [2013?]).

Existem atualmente dois modelos do *Raspberry Pi* no mercado, eles são denominados A e B. Ambos os modelos contam com um processador *Advanced Risc Machine* (ARM) de 700 *Mega Hertz* (MHz), uma unidade de processamento gráfico Videocore IV, saída de vídeo *High-Definition Multimedia Interface* (HDMI) e vídeo composto, saída de áudio, conector de alimentação *Universal Serial Bus* (USB), uma entrada para cartão de memória que armazena o sistema operacional e programas, conector para *flat-cables* de 15 vias para câmeras de vídeo e conexões *General Purpose Input/Output* (GPIO), que são pinos de conexão programáveis, cujo comportamento pode ser controlado e definido via *software*. Tudo isso em dimensões mínimas 85,60 mm x 56 mm e pesando apenas 45 gramas (NETTO, 2013).

A diferença básica entre os modelos é que o A é mais simples, não possui conexão de rede, conta com apenas uma conexão USB e possui 256 *Megabyte* (MB) de memória *Random Access Memory* (RAM), esse modelo é vendido por US$ 25,00. Já o modelo B (figura 1) é um pouco mais completo, ele possui duas conexões USB, conta com uma entrada de rede do tipo *Ethernet* e 512 MB de memória RAM e pode ser comprado por US$ 35,00 (NETTO, 2013).

Figura 1 - *Raspberry Pi* modelo B



Fonte: O autor.

2.2.1 Aplicação

Este pequeno computador pode ser utilizado nos mais diferentes tipos de projetos. A ideia inicial previa sua utilização apenas para fins educacionais, porém como o interesse pela placa foi tanta acabou se abrindo um infinito mundo de possibilidades, com uma gama imensa de projetos e aplicações.

Ele pode ser utilizado como um computador normal ou um computador para programação, seja em Python, C/C++, Java ou até mesmo Assembly. As possibilidades que os pinos GPIO oferecem também são enormes e fazem com que o *Raspberry Pi* também possa ser usado como controlador de um robô ou em soluções que envolvem automação (NETTO, 2013).

Neste trabalho o *Raspberry Pi* é o controlador do sistema de automação residencial, ele juntamente com uma placa de expansão, que por sua vez comunicará diretamente com os equipamentos da residência. Foi optado pelo seu uso devido a ele ter as características necessárias para utilização em um sistema de automação residencial e também por ele ser pouco explorado neste ramo.

2.2.2 *Raspbian*

O *Raspberry Pi* assim como qualquer outro computador precisa de um sistema operacional para funcionar. O *Raspbian* é um dos sistemas operacionais disponíveis para ele, sendo também o sistema operacional mais indicado para o uso. Ele é completo, possui uma interface gráfica, navegador de internet, entre outras funcionalidades. (NETTO, 2013). Como os demais sistemas que podem rodar no *Raspberry Pi* é baseado em Linux, para ser mais específico essa versão em Debian. É um sistema operacional com o kernel otimizado e reduzido, de modo a obter uma melhor performance e utilização do *hardware*.

2.2.3 *General Purpose Input/Output* (GPIO)

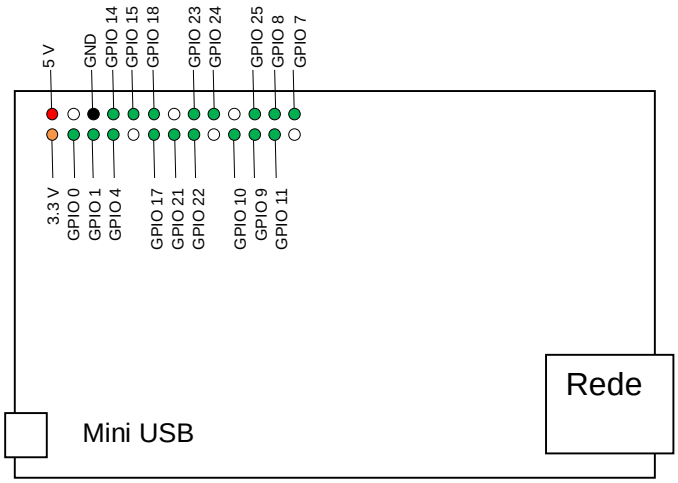
O *Raspberry Pi* pode ser integrado com outros *hardwares* graças as suas conexões GPIO, essas conexões são utilizadas diretamente neste trabalho, para controlar os diferentes tipos de equipamentos conectados ao sistema de automação.

Em português pode-se dizer que *General Purpose Input/Output* (GPIO) são pinos de propósito geral, isso porque fica a cargo do programador decidir se determinado pino será uma entrada ou uma saída de dados e qual será sua função (NETTO, 2013).

O *Raspberry Pi* conta com 26 pinos GPIO, sendo que 17 deles podem ser programados, os demais são pinos de energia, aterramento ou reservados para uso futuro, conforme pode ser observado na figura 2. São esses pinos que permitem que o *Raspberry Pi* seja utilizado no desenvolvimento de sistemas de automação, lendo estados e realizando acionamentos (NETTO, 2013).

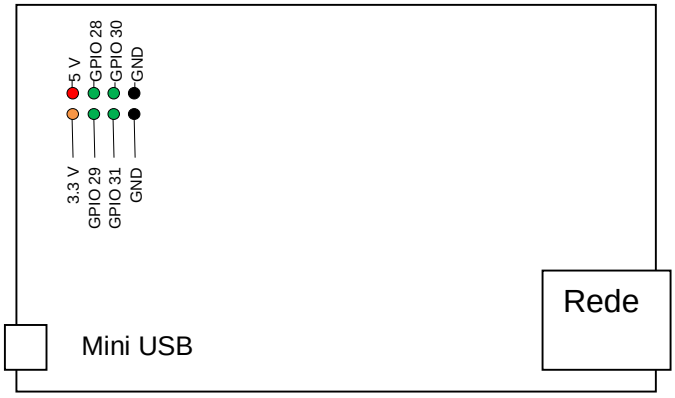
Além desses 26 pinos GPIO o *Raspberry Pi* modelo B conta com mais 8 pinos extras, destes 4 podem ser programados, eles não vêm prontos para uso, mas basta soldar os conectores para utilizá-los. O posicionamento desses pinos pode ser observado na figura 3 (SILVEIRA, [2013?]).

Figura 2 - Organização dos pinos GPIO do *Raspberry Pi* (26 pinos)



Fonte: Adaptado de MONK (2013, p. 4)

Figura 3 - Organização dos pinos GPIO do *Raspberry Pi* (8 pinos)



Fonte: Adaptado de Monk (2013, p. 4) e Silveira ([2013?]).

2.3 PLACAS DE EXPANSÃO

Os pinos GPIO permitiram que o *Raspberry Pi* comunica-se com outros *hardwares*. Para facilitar as integrações começaram as surgir as placas de expansão ou em inglês *Expansion Boards.* Essas placas adicionam ainda mais possibilidades de uso ao *Raspberry Pi*. Atualmente existem inúmeros modelos e variedades, desenvolvidas por diferentes profissionais espalhados pelo mundo.

Essas placas de expansão permitem o desenvolvimento de projetos mais rápidos, e até mesmo por pessoas com pouco conhecimento em eletrônica, uma vez

que, se existir uma placa de expansão com as funcionalidades necessárias, não será preciso investir em tempo e conhecimento para o desenvolvimento de um circuito mais complexo.

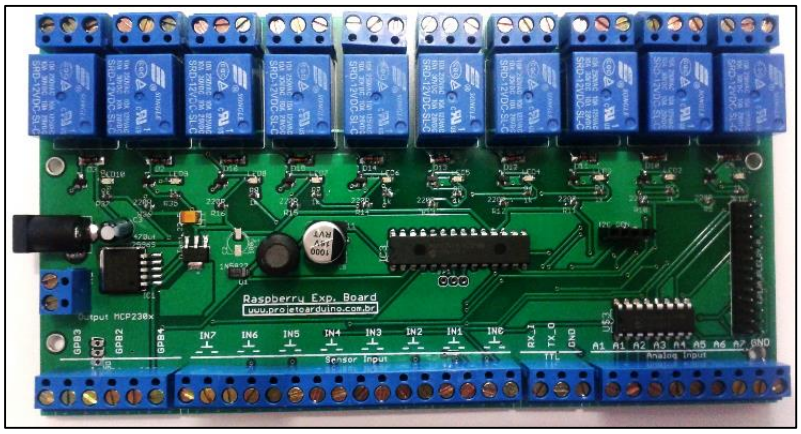
Entre as mais diversas variedades destacam-se as placas para desenvolvimento de automações, como exemplo podemos citar a placa da figura 4, que é utilizada neste trabalho e foi desenvolvida pelo Projeto Arduino, ela permite tanto o desenvolvimento de automação residencial quanto industrial.

Entre as características dessa placa, podemos dizer que ela consegue alimentar o *Raspberry Pi* eliminando assim uma fonte de energia, possui 10 atuadores (relés), 8 entradas para chaves ou sensores que permitem conectar sensores magnéticos, de movimento ou incêndio, 8 entradas analógicas que por padrão o *Raspberry Pi* não tem e 3 transistores para controle de motores ou outros dispositivos que precisam de uma resposta mais rápida. (PROJETO ARDUINO, [2013?]).

Como essa placa é conectada ao *Raspberry Pi* que por sua vez roda Linux, o projeto de automação pode ser baseado em diferentes linguagens como: C, Java, Python, Pearl, entre outras. Além do mais, essa placa é compatível com a biblioteca desenvolvida pela Adafruit (Empresa que também fabrica placas de expansão para o *Raspberry Pi*), essa biblioteca é escrita em Python e tem seu código aberto, podendo ser utilizada para controlar todas as funções da placa de expansão do Projeto Arduino. (PROJETO ARDUINO, [2013?]; RASPBERRY..., [2013?]).

A utilização dessa placa no trabalho foi necessária para evitar a construção de um circuito elétrico próprio, isso acontece devido a necessidade da utilização de alguns componentes eletrônicos para acionar os equipamentos que o *Raspberry Pi* não tem, como relés, e também por permitir a conexão de uma quantidade maior de equipamentos, gerenciados por micro controladores específicos que também não estão presentes no *Raspberry Pi*. A opção por utilizar esse modelo de placa de expansão também foi devido a facilidade de uso e de integração com a linguagem de programação Python.

Figura 4 – Placa para automação residencial desenvolvida pelo Projeto Arduino



Fonte: O autor.

2.4 MOBILIDADE

Como este trabalho utiliza diretamente dispositivos móveis é importante definirmos do que eles se tratam. A mobilidade tem ganhado um enfoque cada vez maior, principalmente pela popularização dos celulares, *notebooks* e *smartphones.*

> [...] mobilidade pode ser definida como a capacidade de acessar informações a partir de qualquer lugar, a qualquer hora. Para isso, é preciso contar com dispositivos móveis como notebooks, laptops, handhelds, entre outros, e uma conexão wireless eficiente, para o acesso aos sistemas remotos e à internet (INTEL, [200-?], p. 1).

Engana-se quem pensa que mobilidade é um termo moderno. O desejo da humanidade comunicar-se a distância e sem a necessidade de fios é bem antigo. Um grande passo foi dado em 1794 com a criação do primeiro telegrafo ótico e bem mais tarde, mais precisamente em 1947 com a criação do conceito de comunicação móvel, ou seja, o celular que começou a operar apenas 20 anos depois (INTEL, [200-?]).

Outro lançamento marcante ocorreu em 1993, quando a Apple lançou o *Apple Newton Message Pad*, o primeiro *Personal Digital Assistant* (PDA) do mercado mundial. No mesmo ano também foi lançado o *Zoomer* PDA desenvolvido pela Palm Computing. Inicialmente nenhum dos modelos fez o sucesso esperado, talvez o preço tenha contribuído, mas apesar disso, eles certamente marcaram o início de uma cadeia evolutiva (INTEL, [200-?]).

Com o passar dos anos, novos modelos de celulares e *smartphones* foram lançados, cada vez com mais recursos e funcionalidades, como câmera digital, acesso

à internet e tela sensível ao toque. Novas empresas também entraram no ramo, e foi em 2007 que o mercado reviveu grandes lançamentos com o Iphone da Apple que fez um grande sucesso e também com o lançamento da plataforma Android do Google, que por sua vez não chamou tanta atenção quanto o lançamento do Iphone. Mas, alguns anos depois podemos dizer que o Android se tornou um sucesso, sendo uma plataforma mundialmente conhecida e famosa, com milhares de aplicativos à disposição dos usuários (KAWAMOTO, 2011).

Atualmente a popularização dos dispositivos móveis tem proporcionado o acesso a essas tecnologias por milhares de pessoas, essa popularização acaba proporcionando a exploração de uma área em constante evolução e contribui para o surgimento de novos aplicativos, ferramentas e tecnologias que tendem a facilitar nosso dia-a-dia.

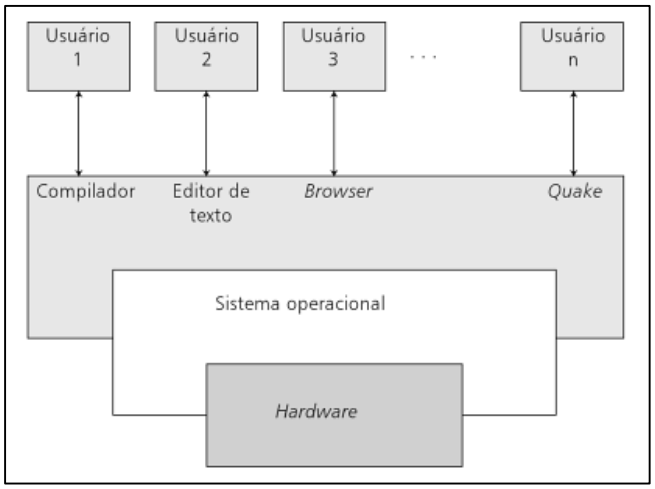
2.5 SISTEMAS OPERACIONAIS

Tanto dispositivos móveis quanto computadores precisam de um sistema operacional para funcionarem, é por isso que eles são tão importantes no mundo da tecnologia da informação, de nada adianta você ter uma ótima máquina sem um bom sistema operacional, e de nada adianta você ter um sistema operacional sem ter onde executá-lo.

Sistema operacional nada mais é que a camada de *software* colocada entre o *hardware* e os programas que o usuário executa (figura 5) (OLIVEIRA; CARISSIMI; TOSCANI, 2010). Ele é o responsável pela comunicação com o *hardware,* isto é, faz a ligação dos *softwares* do usuário com os periféricos, além da distribuição e controle dos recursos computacionais. Segundo Oliveira, Carissimi e Toscani (2010):

> O sistema operacional é responsável pelo acesso aos periféricos; sempre que algum programa necessita de algum tipo de operação de entrada e saída, ele solicita ao sistema operacional. Dessa forma o programador não precisa conhecer os detalhes do *hardware.* Informações do tipo “como enviar um caractere para impressora” ficam escondidas dentro do sistema operacional. Ao mesmo tempo, como todos os acessos aos periféricos são feitos através do sistema operacional, ele pode controlar qual programa está acessando qual recurso. É possível, então, obter uma distribuição justa e eficiente dos recursos. (OLIVEIRA; CARISSIMI; TOSCANI, 2010, p. 22-23).

Figura 5 - Sistema Computacional



Fonte: OLIVEIRA, CARISSIMI, TOSCANI (2010, p. 22)

O sistema operacional procura tornar a utilização do computador, mais eficiente e conveniente (OLIVEIRA; CARISSIMI; TOSCANI, 2010). Eles existem a mais de 50 anos, e durante todo esse tempo uma grande variedade foi desenvolvida, mas nem todos são muito conhecidos (TANENBAUM, 2009).

Entre os principais sistemas operacionais destacam-se o Windows, o Linux e o Mac OS. Como apenas o Linux foi utilizado diretamente neste trabalho, conheceremos um pouco mais sobre ele na sequência.

2.5.1 Linux

O Linux é um sistema operacional completo, desenvolvido pelo finlandês Linus Torvalds. Um clone do sistema Unix, sua primeira versão a 0.01 foi liberada em 1991, essa versão continha 9.300 linhas de códigos escritos em C e mais 950 linhas em linguagem Assembly (TANENBAUM, 2009).

Segundo Campos (2006, p.1) “Linux é ao mesmo tempo um kernel (ou núcleo) e o sistema operacional que roda sobre ele, dependendo do contexto em que você encontra a referência”. Atualmente o sistema é mantido por uma comunidade de desenvolvedores composta por pessoas e entidades coordenada por Linus.

Um sistema operacional de código aberto, livre que adota uma licença do tipo *General Public License* (GPL), o que significa que todos os interessados podem utilizá-

lo e redistribuí-lo nos devidos termos de licença. Aliado com outros *softwares* livres o Linux pode formar um ambiente moderno, seguro, eficiente e estável para computadores em geral, servidores e sistemas embarcados (CAMPOS, 2006).

O uso do Linux neste trabalho acontece diretamente no *Raspberry Pi*, porém outro sistema operacional derivado do mesmo é utilizado e tratado a seguir, o Android.

2.6 ANDROID

A plataforma Android vem crescendo muito nos últimos, tanto em número de usuários quanto em funcionalidades. Esse foi o principal fator que levou essa plataforma a ser escolhida para o desenvolvimento móvel deste trabalho.

A plataforma Android foi desenvolvida com base no sistema operacional Linux, com objetivo de permitir que os desenvolvedores criem aplicações móveis que tirem o máximo de proveito de seus aparelhos. É um projeto de código aberto, em constante evolução, que oferece um conjunto completo de *softwares* para dispositivos móveis (OHA, [200-?]).

“O Android é uma plataforma para tecnologia móvel completa, envolvendo um pacote com programas para celulares, já com um sistema operacional *middleware*, aplicativos e interface do usuário.” (PEREIRA; SILVA, 2009, p. 3).

Por ele ser *open source*, isto é, ter seu código aberto, o mesmo tem atraído a atenção de muitos fabricantes de aparelhos telefônicos, uma vez que o mesmo pode ser modificado e adaptado conforme as suas necessidades (PEREIRA; SILVA, 2009).

Por trás de seu desenvolvimento temos o Google e também um grupo formado por empresas de telefônica móvel, fabricantes de aparelhos e desenvolvedores de *software*. Criado para padronizar uma plataforma móvel de código aberto, a fim de atender as necessidades do mercado, esse grupo é conhecido como *Open Handset Alliance* (OHA). (LECHETA, 2013).

Lecheta (2013) comenta que toda a segurança do Android é baseada na segurança do Linux, cada aplicação é executada em um processo separado, que possuí uma *thread* dedicada e tem seu próprio usuário no sistema operacional, portanto nenhuma aplicação pode ter acesso aos diretórios e arquivos de outra. Está

é uma característica bastante interessante que reforça a segurança do sistema operacional.

2.7 LINGUAGENS DE PROGRAMAÇÃO

Linguagens de programação podem ser definidas como um conjunto de instruções (vocabulário) associado a um conjunto de regras (sintaxe). Elas são usadas no desenvolvimento de programas para resolução de um determinado problema (WILLRICH, [200-?]). Esses programas podem rodar em diferentes plataformas e *hardwares*, sejam eles computadores ou dispositivos móveis.

A fim de construir o sistema de automação residencial, foram utilizadas duas linguagens de programação distintas neste trabalho. A seguir, será apresentada as principais características de cada uma delas.

2.7.1 Java

Java é uma linguagem de programação orientada a objetos, que roda independente de plataformas. Teve o início de seu desenvolvimento em 1991, quando um grupo de desenvolvedores da Sun Microsystems liderado por James Gosling estava trabalhando no desenvolvimento de um projeto de TV interativa, James não satisfeito com os resultados da linguagem utilizada, trancou-se em um escritório e criou a nova linguagem baseada em C e C++. O Java propriamente dito foi lançado pela Sun Microsystems no segundo semestre de 1995, em um kit de desenvolvimento gratuito. (CADENHEAD; LEMAY, 2005).

O Java é uma linguagem compilada que tornou-se popular justamente por rodar em múltiplas plataformas, por ser totalmente orientada a objetos, eficiente, possuir suporte a concorrência com o uso de *threads* e também por ser uma linguagem segura, devido a sua coleta automática de lixo que evita erros comuns dos programadores e por seus tratamentos de exceção, que permitem à aplicação continuar rodando mesmo em condições anormais. (CESTA, 2009).

O Java foi utilizado no desenvolvimento do aplicativo para Android, possibilitando o controle da automação pelo usuário, de maneira fácil e intuitiva, de onde quer que ele esteja. Esta linguagem foi escolhida por ser a nativa da plataforma Android, facilitando assim o acesso aos recursos do sistema.

2.7.2 Python

Python é uma linguagem de programação de auto nível, orientada a objetos, de tipagem dinâmica e forte, interpretada e interativa. Possui uma sintaxe clara e concisa que favorece a legibilidade do código e torna a linguagem mais produtiva (BORGES, 2010).

A linguagem foi criada em 1990 por Guido Van Rossum na Holanda. A mesma tinha foco em usuários como físicos e matemáticos e foi concebida a partir de uma outra linguagem existente na época conhecida como ABC. Atualmente a linguagem é bem aceita na indústria e por empresas de alta tecnologia. (BORGES, 2010).

O Python possui diversas estruturas de alto nível, uma vasta coleção de módulos prontos para uso, além de diversos *frameworks* de terceiros. Esta linguagem também possui recursos encontrados em linguagens mais modernas tais como: geradores, introspecção, persistência, metaclasses e unidades de teste. O Python é interpretado através de *bytecode* pela máquina virtual Python, tornando o código portável, ou seja, possível de compilar aplicações em uma plataforma e rodar em outra, ou executar direto do código fonte. (BORGES, 2010).

Python é uma linguagem de código aberto, com uma licença compatível com a GPL, porém menos restritiva, pois permite que o mesmo seja incorporado inclusive em produtos proprietários. Atualmente a linguagem é mantida pela Python Software Foundation. Além da aplicação como linguagem principal também é possível utilizá-lo como linguagem *script* em vários *softwares,* ou integrá-lo com outras linguagens como C e Fortran. (BORGES, 2010).

Neste trabalho, o Python foi utilizado para o desenvolvimento do aplicativo servidor, que é executado no *Raspberry Pi.* Esse aplicativo é fundamental, ele recebe os comandos enviados pelo aplicativo móvel e executa os mesmos na residência, ou seja, ele é o responsável pela comunicação com o Android e também pela

comunicação com o *hardware* envolvido. Essa linguagem foi escolhida, pelo fato de ser a mais indicada para o uso com o *Raspberry Pi* e também por facilitar a comunicação com os pinos GPIO do mesmo.

2.8 REDES DE COMPUTADORES

A fim de estabelecer a comunicação entre o dispositivo móvel e o *Raspberry Pi* é necessário a utilização de uma rede de computadores.

Segundo Nascimento (2011, p.4) “Uma rede de computadores é formada por um conjunto de módulos processadores capazes de trocar informações e compartilhar recursos, interligados por um sistema de comunicação (meios de transmissão e protocolos)”.

As redes são de extrema importância, tanto em ambientes residenciais como empresariais, pois propiciam inúmeros benefícios como compartilhamento de recursos, impressoras, arquivos, entre outros. Além de ser um ótimo meio de comunicação através da internet (NASCIMENTO, 2011).

Entre as diversas classificações das redes de computadores encontra-se as redes locais. Essas redes geralmente são privadas e permitem a interconexão de dispositivos em uma pequena região. As redes locais ou *Local Area Networks* (LANs) como são conhecidas, podem ser cabeadas, sem fios ou mistas (NASCIMENTO, 2011).

As LANs cabeadas mais comuns utilizam o padrão imposto pelo *Institute of Electrical and Electronics Engineers* (IEEE), conhecido como IEEE 802.3 podendo atingir velocidades de 100 *megabit* por segundo (Mbps) à 10 *gigabit* por segundo (Gbps). Outro padrão bastante popular e geralmente encontrado em *notebooks* e *smartphones* é o IEEE 802.11 conhecido como *Wi-Fi* ou *Wireless Local Area Netwok* (WLAN), esse padrão pode alcançar uma velocidade de até 300 Mbps (NASCIMENTO, 2011).

As redes *Wi-Fi* estão se tornando cada vez mais populares e utilizadas, pois permitem conexões de qualidade sem o uso de fios. Tanto o padrão IEEE 802.3 quanto o IEEE 802.11 podem ser utilizadas no sistema proposto por este trabalho.

2.9 UNIFIED MODELING LANGUAGE (UML)

Antes de iniciar o desenvolvimento de qualquer *software* é extremamente importante analisar cada processo e modelar como o sistema deve se comportar.

Unifield Modeling Language (UML) ou Linguagem de Modelagem Unificada, é uma linguagem visual utilizada para modelar sistemas por meio do paradigma da orientação a objetos. Essa linguagem tornou-se padrão na modelagem de sistemas e é internacionalmente usada pela engenharia de software (GUEDES, 2007).

Segundo Guedes (2007) cada diagrama da UML analisa o sistema ou parte dele, entre os principais tipos de diagramas podemos citar:

- Diagramas de casos de uso: é o diagrama mais geral e informal da UML, utilizado principalmente para auxiliar no levantamento e análise dos requisitos.
- Diagrama de classes: é o diagrama mais utilizado e importante da UML, este diagrama define a estrutura das classes utilizadas pelo sistema, além dos atributos, métodos e relacionamentos.
- Diagrama de objetos: é um complemento ao diagrama de classes, fornece uma visão dos valores armazenados pelos objetos de um diagrama de classes em um determinado momento.
- Diagrama de sequência: preocupa-se com a ordem temporal em que as mensagens são trocadas entre os objetos envolvidos em um determinado processo.

Além dos diagramas descritos acima podemos citar outros como: diagramas de comunicação, estrutura composta, máquina de estados, atividade, interação geral, componentes, implantação, pacotes e diagramas de tempo (GUEDES, 2007).

3 CAMPO OU ÁREA DE ESTUDO

Nos últimos anos a automação residencial ganhou novos adeptos e tecnologias, isso ocasionou um crescimento significativo nas residências automatizadas. É por esses fatores que a área de estudo deste trabalho concentrou-se no desenvolvimento de uma solução para controle de automação residencial baseada em novas plataformas, mais baratas, populares e pouco exploradas. Essa solução englobou desenvolvimento móvel e desktop, além de integrações com *hardwares* e circuitos elétricos.

O campo de estudo concentrou-se em pessoas comuns que procuram um controle mais moderno, centralizado, eficiente e automatizado de suas residências com menor custo de implantação, além de entusiastas, estudantes e pessoas já ligadas ao ramo da automação.

A homologação ou teste deste trabalho foi possível através do desenvolvimento de um protótipo, onde foram aplicadas as soluções propostas e desenvolvidas no decorrer do mesmo.

4 MÉTODOS

Este capítulo apresenta a classificação quanto aos métodos, técnicas e perguntas de pesquisa do trabalho.

4.1 DELIMITAÇÃO DO ESTUDO, MÉTODOS E TÉCNICAS DE COLETA DE DADOS

Este trabalho caracteriza-se por ser uma pesquisa exploratória, a mesma objetivou-se no desenvolvimento de uma solução para controle de automação residencial a fim de possibilitar aos seus utilizadores um controle à distância ou centralizado de suas residências. As pesquisas realizadas a fim de esclarecer os diversos pontos expostos nesse trabalho foram realizadas em livros, sites, artigos e outras fontes confiáveis.

O desenvolvimento do trabalho foi realizado entre os anos de 2013 e 2014 pelo acadêmico Rodrigo Batistello orientado pelo professor Jose Luiz Cunha Quevedo como requisito parcial para obtenção de grau no curso de Sistemas de Informação na Universidade do Oeste de Santa Catarina – Unoesc, unidade Chapecó que fica localizada na Av. Nereu Ramos 3777-D, bairro Seminário, CEP 89813-000.

Quanto aos métodos utilizados este trabalho caracteriza-se por possuir uma abordagem dialética, pois busca argumentar e contra argumentar sobre o assunto e tem um procedimento monográfico, pois foi realizada uma pesquisa aprofundada sobre o assunto.

Quantos as técnicas foram utilizadas documentação indireta, tanto pesquisa documental, como sites da web, quanto pesquisas bibliografias, livros, revistas, monografias, entre outros.

4.2 CARACTERIZAÇÃO DO ESTUDO

Quanto aos fins, o estudo caracteriza-se por exploratório, aplicado e descritivo devido ao levantamento bibliográfico e a possibilidade de experiências diretas na solução proposta, tanto na parte de *software* com o aplicativo, quanto na parte de *hardware* com a criação do protótipo para exemplificação.

Quanto aos meios, caracteriza-se em bibliográfica devido a fundamentação teórica ser extraída de fontes confiáveis como livros, revistas artigos e monografias. E também como experimental, devido ao desenvolvimento do protótipo, que possibilita o controle de iluminação, eletroeletrônicos, alarme, entre outros.

4.3 TÉCNICA DE ANÁLISE E INTERPRETAÇÃO DE DADOS

A técnica de análise do trabalho é qualitativa, os resultados do mesmo foram apresentados através de documentos, figuras e de um protótipo de automação completo, composto pelos aplicativos necessários e de toda parte de *hardware* fundamental.

4.4 QUESTÕES DE PESQUISA

- Quais são os equipamentos, componentes eletrônicos e ferramentas que melhor se adaptam ao trabalho?
- O *hardware* utilizado é suficiente e adequado para execução das tarefas propostas?

5 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Este capítulo tem como objetivo apresentar o desenvolvimento do trabalho afim de atingir os objetivos propostos. Será apresentado o protótipo construído por meio de explicações, imagens, quadros, capturas de telas do sistema desenvolvido e toda a parte de modelagem criada.

5.1 MODELAGEM DO SISTEMA

O sistema é composto por dois aplicativos desenvolvidos em diferentes linguagens de programação. O aplicativo servidor foi desenvolvido em Python e o aplicativo cliente foi desenvolvido em Java.

Antes de iniciar o desenvolvimento foi feita toda a análise necessária, nesta análise foi levantado todos os requisitos funcionais e não funcionais, foi criado e expandido o caso de uso além do modelo entidade-relacionamento (MER) para representar o banco de dados sob o qual o sistema é executado. Na sequência será detalhado cada um desses itens.

5.1.1 Requisitos funcionais e não funcionais

Todo desenvolvimento de *software* exige o levantamento de requisitos, sejam eles funcionais ou não funcionais. Para o desenvolvimento deste trabalho foram levantados os requisitos presentes no quadro 1.

Quadro 1 - Requisitos funcionais e não funcionais

1. Controlar luzes e outros equipamentos		
Descrição: Deverá ser possível o controle de luzes e outros equipamentos eletrônicos conectados a relés no sistema de automação, permitindo ligar e desligar estes remotamente.		
Requisitos funcionais		
Nome	Restrições	Categoria

RF 1.1 Identificar os equipamentos	Deverá ser possível a identificação do equipamento no sistema de automação através de um nome definido pelo usuário.	Interface
RF 1. 2 Ligar e desligar equipamentos	Deve ser possível ligar e/ou desligar determinado equipamento remotamente.	Usabilidade
RF 1.3 Identificar equipamentos ligados/desligados	Devem ser identificados os equipamentos ou luzes ligados e desligados no momento.	Interface
2. Controlar sistema de alarme		
Descrição: O *software* deve permitir o controle e o monitoramento do sistema de alarme residencial remotamente.		
Requisitos funcionais		
Nome	Restrições	Categoria
RF 2.1 Ligar e desligar alarme	Deve ser possível ligar e desligar o alarme remotamente pelo usuário a qualquer momento.	Usabilidade
RF 2.2 Ligar e desligar pânico	Permitir o disparo imediato da sirene pelo usuário, bem como o desligamento da mesma quando desejado.	Usabilidade
RF 2.3 Monitorar status	Permitir controlar o status do alarme bem como visualizar os últimos disparos ocorridos.	Interface
RF 2.4 Gerenciar sensores	Deve ser possível renomear os sensores e configurar quais sensores estão ativos no momento.	Usabilidade Interface
RF 2.5 Configurar opções de disparo	Permitir selecionar se a sirene deve ser disparada ao detectar algum movimento nos sensores e por quanto tempo a mesma deve permanecer ativa.	Usabilidade
RF 2.6 Configurar envio de e-mail	Deve ser possível configurar um endereço de e-mail para receber a notificação de alarme disparado.	Usabilidade
3. Temperatura e humidade		
Descrição: Permitir através de uma interface a visualização da temperatura e da humidade da residência.		
Requisitos funcionais		
Nome	Restrições	Categoria
RF 3.1 Visualizar temperatura e humidade	Permitir a visualização da temperatura e humidade da residência, podendo atualizar os dados a qualquer momento.	Interface
4. Câmera de segurança		
Descrição: Utilizada para monitoramento da residência remotamente.		
Requisitos funcionais		
Nome	Restrições	Categoria
RF 4.1 Permitir a visualização da câmera	Criar uma interface onde o usuário possa visualizar uma câmera instalada na residência e conectada no sistema de automação a qualquer momento.	Interface
5. Manter agendamentos		
Descrição: Utilizado para cadastrar agendamentos no sistema, bem como a visualização dos agendamentos cadastrados ou a remoção dos mesmos.		
Requisitos funcionais		
Nome	Restrições	Categoria
RF 5.1 Cadastrar agendamentos	O sistema deve permitir o cadastro de agendamentos, informando um nome, os equipamentos que deve ser acionados, a data e hora para ligar e desligar e os dias da semana que deve ser repetido.	Usabilidade Interface
RF 5.2 Remover agendamento	O sistema deve permitir a exclusão de um agendamento para que o mesmo não seja mais executado.	Usabilidade

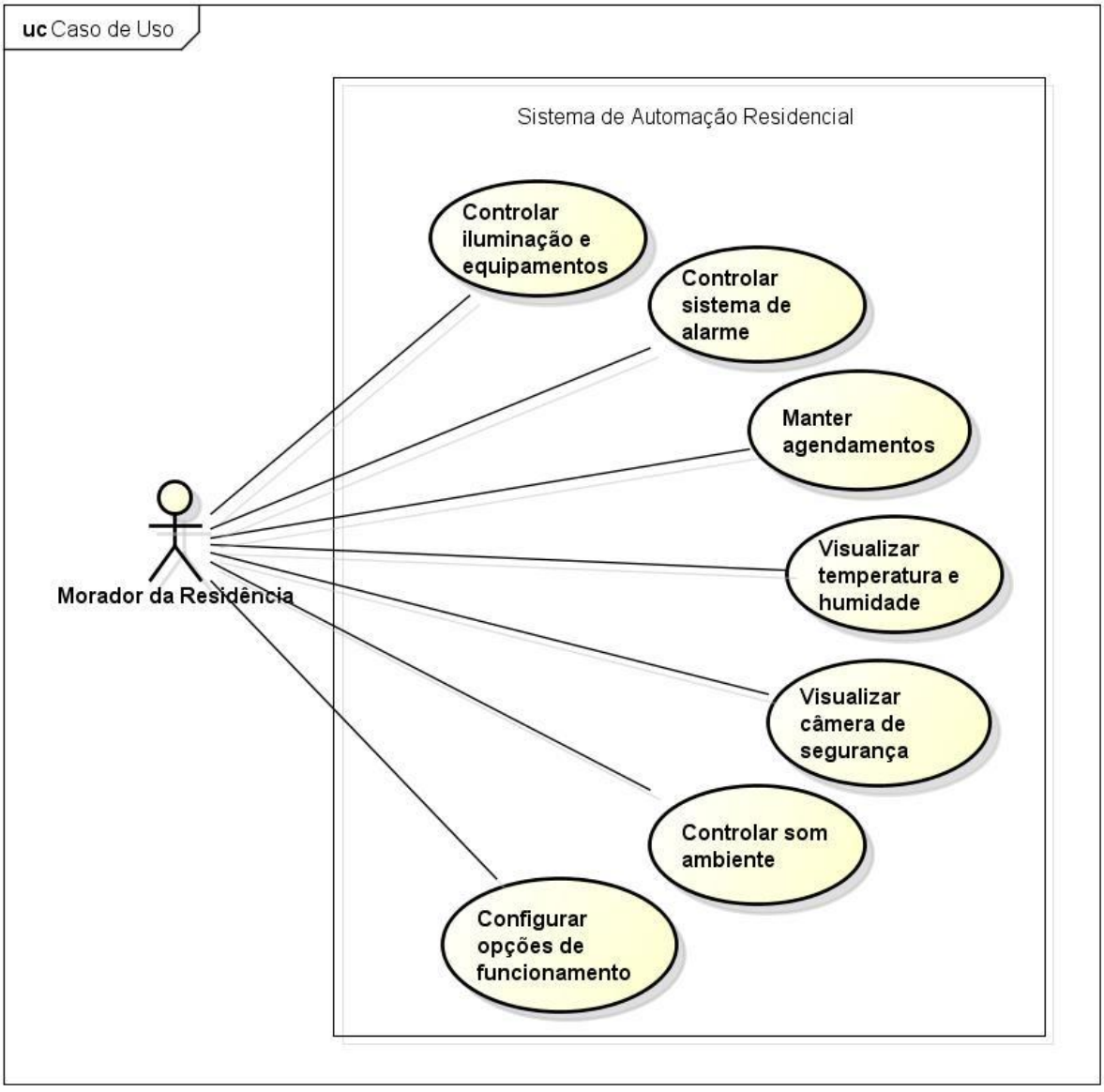
RF 5.3 Consulta de agendamento	Permitir a consulta de agendamentos no sistema.	Interface
Requisitos não funcionais		
Nome	Restrição	Categoria
RNF 5.4 Desativar automaticamente	O agendamento deve ser desativado automaticamente após sua execução, sendo assim o mesmo não deve mais aparecer consulta de agendamento.	Interface Segurança
6. Controlar som ambiente		
Descrição: Permitir o controle remotamente de um sistema de som na residência.		
Requisitos funcionais		
Nome	Restrições	Categoria
RF 6.1 Executar	Permitir a execução de uma música selecionada pelo usuário.	Usabilidade
RF 6.2 Pausar	Permitir que o usuário pause e retorne a execução da música a qualquer momento.	Usabilidade
RF 6.3 Avançar e retroceder	Permitir avançar para a próxima faixa musical ou retroceder para a faixa anterior.	Usabilidade
RF 6.4 Listar músicas	Exibir a lista de músicas armazenadas no sistema, bem como uma busca por nome da faixa.	Usabilidade Interface
Requisitos não funcionais		
Nome	Restrição	Categoria
RNF 6.5 Armazenamento	As músicas devem ser armazenadas pelo usuário em uma pasta disponível através de acesso ao *File Transfer Protocol* (FTP) do sistema, esse acesso deve ser feito através de um computador.	Segurança
7. Instalação e utilização		
Descrição: Requisitos necessários para utilização do sistema bem como sua instalação.		
Requisitos não funcionais		
Nome	Restrições	Categoria
RNF 7.1 *Raspberry Pi*	O servidor deve rodar exclusivamente no *Raspberry Pi* conectado a uma placa de automação especifica.	Sistema Operacional *Hardware*
RNF 7.2 Android	O aplicativo cliente roda em plataforma Android 2.3.6 ou superior.	Sistema Operacional
RNF 7.3 Banco de dados	O banco de dados do servidor deve ser MySQL.	Segurança
RNF 7.4 Login	O sistema deve contar com um login e senha de proteção.	Segurança
RNF 7.5 Conexão	Para utilização é necessária conexão de rede entre o servidor e o cliente, além de conexão com a internet para envio do e-mail no disparo do alarme.	Rede
RNF 7.6 Múltiplas conexões	O sistema deve permitir múltiplas conexões	Paralelismo
RNF 7.7 Auto recuperação	O sistema deve restaurar o estado anterior no caso de um desligamento inesperado.	Confiabilidade
RNF 7.8 Velocidade	O sistema deve responder a qualquer comando em menos de 15 segundos.	Velocidade

Fonte: O autor.

5.1.2 Caso de uso

Para proporcionar um melhor entendimento do sistema também foi elaborado um diagrama de caso de uso. O mesmo pode ser visualizado na figura 6.

Figura 6 - Caso de uso



Fonte: O Autor

5.1.2.1 Expansão do caso de uso

A fim de explicar cada caso de uso foi realizada a expansão do mesmo, assim é possível entende-lo ainda melhor. Esta expansão está descrita no quadro 2.

Quadro 2 - Expansão do caso de uso

Controlar iluminação e outros equipamentos
Ator: Morador da residência *1. Descrição:* Ligar ou desligar um equipamento ou iluminação. *2. Fluxo:* 2.1 [IN] O morador da residência informa a opção. 2.2 [OUT] O sistema exibe os botões para controle dos equipamentos. 2.3 [IN] O morador escolhe um equipamento para ligar ou desligar. 2.4 [OUT] O sistema executa o comando e apresenta o resultado alterando a cor do botão. *3. Exceções* 3.1 Exceção 2.4a: Erro na execução do comando 2.4a.1 Retornar ao passo 2.3
Controlar sistema de alarme
Ator: Morador da residência *1. Descrição:* Ligar, desligar e monitorar o sistema de alarme. *2. Fluxo:* 2.1 [IN] O morador escolhe a opção para controle do alarme. 2.2 [OUT] O sistema exibe as opções disponíveis (Ligar, desligar alarme e pânico ou últimos disparos). 2.3 [IN] O morador escolhe a opção desejada. 2.4 [OUT] O sistema executa a opção selecionada e exibe o retorno. *3. Exceções* 3.1 Exceção 2.4a: Erro na execução do comando 2.4a.1 Retornar ao passo 2.3
Manter agendamentos
Ator: Morador da residência *1. Descrição:* Inserir, remover e filtrar agendamentos. *2. Fluxo* 2.1 [IN] O morador escolhe a opção para manter agendamento. 2.2 [OUT] O sistema exibe uma interface com os agendamentos cadastrados. *3. Variantes* 3.1 Inserir agendamento 3.2 Filtrar agendamento 3.3 Remover agendamento **Variante 3.1 Inserir agendamento** 3.1.1 [IN] O morador escolhe a opção de novo agendamento. 3.1.2 [OUT] O sistema exibe a interface para inserir um novo agendamento. 3.1.3 [IN] O morador preenche as informações conforme desejado (nome, equipamentos, datas e dias da semana). 3.1.4 [IN] O morador salva ou cancela a inserção do agendamento. 3.1.5 [OUT] O sistema exibe o resultado da solicitação. **Variante 3.2 Filtrar agendamento** 3.2.1 [IN] O morador preenche o campo de filtro 3.2.2 [OUT] O sistema exibe os agendamentos que atendem a condição do filtro. **Variante 3.3 Remover agendamento** 3.3.1 [IN] O morador escolhe um agendamento e clica em remover. 3.3.2 [OUT] O sistema exibe o resultado da solicitação de remoção.
Visualizar temperatura e humidade
Ator: Morador da residência *1. Descrição:* Exibe a temperatura e a humidade ambiente.

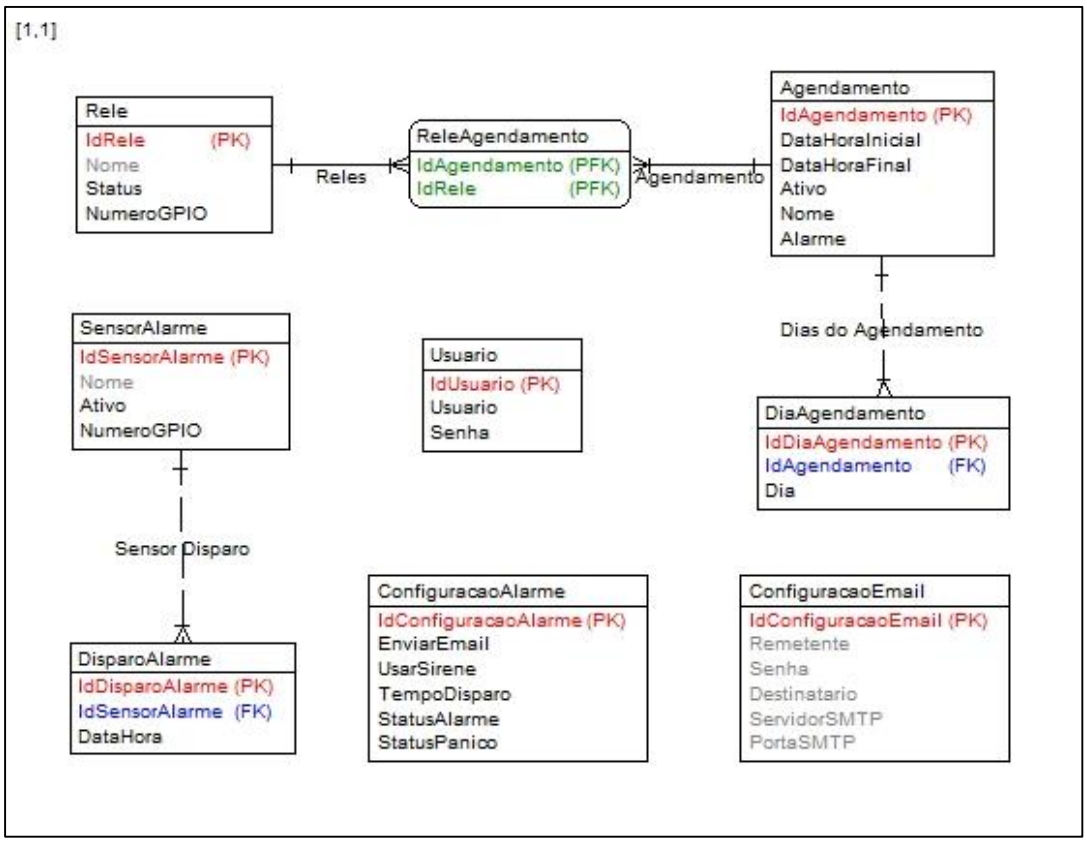
2. Fluxo 2.1 [IN] O morador escolhe a opção para visualizar a temperatura e a humidade. 2.2 [OUT] O sistema retorna os dados para o usuário.
Visualizar câmera de segurança
Ator: Morador da residência *1. Descrição:* Exibe a imagem da câmera de segurança. *2. Fluxo* 2.1 [IN] O morador escolhe a opção de visualização de câmera. 2.2 [OUT] O sistema exibe a imagem da câmera de segurança conectada ao sistema.
Controlar som ambiente
Ator: Morador da residência *1. Descrição:* Controla o som ambiente da residência. *2. Fluxo* 2.1 [IN] O morador da residência escolhe a opção de controle do som ambiente. 2.2 [OUT] O sistema apresenta uma interface com a lista de músicas onde é possível, filtrar, executar, pausar, parar, avançar ou retroceder. *3. Variantes* 3.1 Executar 3.2 Pausar 3.3 Avançar 3.4 Retroceder 3.5 Parar 3.6 Filtrar **Variante 3.1 Executar** 3.1.1 [IN] O morador escolhe uma faixa para reprodução ou clica no botão “Play”. 3.1.2 [OUT] O sistema executa a música desejada. **Variante 3.2 Pausar** 3.2.1 [IN] O morador clica no botão correspondente ao pausar. 3.2.2 [OUT] O sistema pausa a música em execução. **Variante 3.3 Avançar** 3.3.1 [IN] O morador clica no botão correspondente ao avanço de faixa. 3.3.2 [OUT] O sistema avança para a próxima faixa. **Variante 3.4 Retroceder** 3.4.1 [IN] O morador clica no botão correspondente ao retroceder. 3.4.2 [OUT] O sistema volta para a faixa anterior. **Variante 3.5 Parar** 3.5.1 [IN] O morador clica no botão correspondente para parar a execução da música. 3.5.2 [OUT] O sistema para a execução da música. **Variante 3.6 Filtrar** 3.6.1 [IN] O morador preenche o campo de filtro. 3.6.2 [OUT] O sistema filtra as faixas musicais que atendem ao filtro aplicado.
Configurar opções de funcionamento
Ator: Morador da residência *1. Descrição:* Exibe as configurações disponíveis, além de permitir a modificação das mesmas. *2. Fluxo* 2.1 [IN] O morador escolhe a opção de configuração do sistema; 2.2 [OUT] O sistema exibe as configurações disponíveis por aba (geral, relés, alarme, e-mail). 2.3 [IN] O morador escolhe a aba desejada. 2.4 [OUT] O sistema exibe as configurações da aba. 2.5 [IN] O morador faz as alterações conforme desejado, salvar ou cancela as alterações. *3. Variantes* 3.1 Salvar alterações 3.2 Cancelar alterações **Variante 3.1 Salvar alterações** 3.1.1 [IN] O morador salva a nova configuração. 3.1.2 [OUT] O sistema exibe uma mensagem com o resultado da solicitação. **Variante 3.2 Cancelar alterações** 3.2.1 [IN] O morador retorna sem salvar as alterações. 3.2.2 [OUT] O sistema retorna ao estado anterior.

Fonte: O autor.

5.1.3 Modelo Entidade-Relacionamento

O modelo entidade-relacionamento (MER) é utilizado para representação do banco de dados utilizado pelo sistema. Este é simples, composto por apenas 9 tabelas e 4 relacionamentos. Ele foi desenvolvido para utilização no banco de dados MySQL. Foi pensado em um modelo simples para um melhor funcionamento no *hardware* disponível. O mesmo pode ser visualizado na figura 7.

Figura 7 - MER



Fonte: O autor.

A explicação para cada tabela do MER criado é a seguinte:

- **Rele:** Tabela que armazena as configurações dos relés da placa de automação utilizada, essa tabela é preenchida de forma fixa com 13 registros que correspondem a 10 relés e outros 3 transistores. Nessa tabela é possível através do sistema apenas alterar o nome e o status dos relés.
- **SensorAlarme:** Tabela que armazena as informações dos sensores de alarme da placa de automação. Preenchida de forma fixa com 8 registros que

correspondem ao número de sensores da placa. Através do sistema é possível alterar o nome do sensor e o campo Ativo que diz se o mesmo está ou não sendo utilizado.

- **DisparoAlarme:** Tabela que armazena os disparos do alarme, ela está relacionada com a tabela de SensorAlarme, a fim de saber qual sensor originou o disparo. Através da mesma também é possível saber a data e a hora que o mesmo ocorreu.
- **ConfiguracaoAlarme:** Tabela que armazena as configurações definidas pelo usuário, que dizem como o alarme deve se comportar, não tem relacionamento com outras tabelas pois é utilizada em uma função específica no sistema, fazendo-se desnecessária essa relação.
- **ConfiguracaoEmail:** Tabela que armazena a configuração do e-mail feita pelo usuário. Armazena o destinatário do e-mail e os dados do servidor que será utilizado para envio. Não tem relação com outras tabelas pois armazena uma única configuração.
- **Usuário:** Tabela que armazena o usuário e a senha para login no sistema, não é permitido o cadastro de mais de um usuário, é possível apenas alterar o existente.
- **Agendamento:** Tabela que armazena os agendamentos definidos pelo usuário, datas, status, nome.
- **DiaAgendamento:** Tabela que armazena os dias que o agendamento deve ser executado, está relacionada com a tabela de Agendamento através de uma chave estrangeira. Os dias ficam armazenados em um campo inteiro onde domingo corresponde a 0 e sábado a 6.
- **ReleAgendamento:** Tabela que armazena os relés que devem ser acionados para aquele agendamento. Está relacionada através de uma chave primaria e estrangeira com a tabela de Rele e também com a tabela Agendamento.

5.2 INSTALAÇÃO DO SISTEMA OPERACIONAL E APLICATIVOS

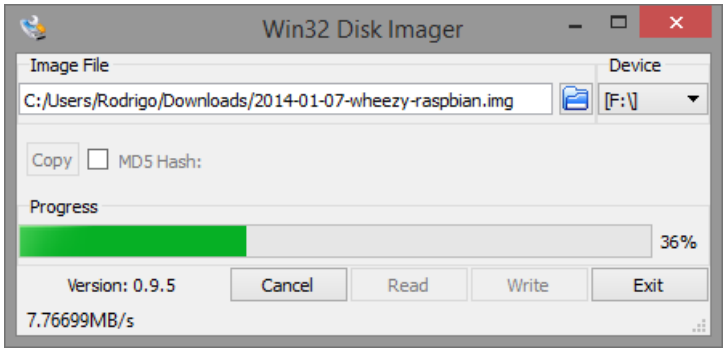
Para o desenvolvimento deste trabalho foi necessário realizar a instalação e configuração de diversas bibliotecas, aplicativos e o próprio sistema operacional do

Raspberry Pi. Como o mesmo utiliza Linux, que é um sistema operacional menos conhecido pelas pessoas as mesmas serão detalhadas na sequência.

5.2.1 Instalação do sistema operacional no *Raspberry Pi*

Como mencionado na fundamentação teórica foi optado pelo uso do sistema operacional *Raspbian*, o mesmo encontra-se disponível para *download* no site oficial do *Raspberry Pi* (http://www.raspberrypi.org/downloads/). Efetuado o *download* foi necessário extrair o arquivo que encontra-se compactado e utilizar a ferramenta *Win32 Disk Imager* (http://sourceforge.net/projects/win32diskimager) para copiar o sistema operacional para o cartão de memória, esse precisa ter pelo menos 4GB livre e uma velocidade igual ou superior a classe 4, esse processo pode ser observado na figura 8.

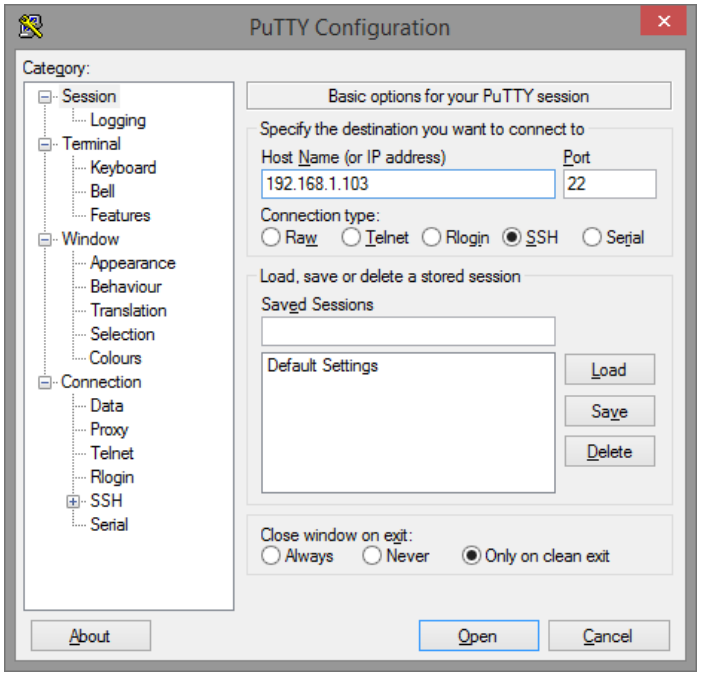
Figura 8 - Envio do sistema operacional para o cartão de memória



Fonte: O autor.

Depois de carregar o sistema operacional no cartão de memória, foi inserido o mesmo no *Raspberry Pi*, conectado o mesmo na rede cabeada e ligado na energia. Verificando as conexões ativas na tabela do roteador foi obtido o endereço IP do mesmo, e assim foi utilizada a ferramenta *Putty* (http://www.putty.org/) para estabelecer uma conexão *Secure Shell* (SSH), que nada mais é que uma conexão remota, assim ficou dispensado o uso de um monitor, teclado e mouse. A interface da ferramenta pode ser observada na figura 9.

Figura 9 - Interface da ferramenta *Putty*

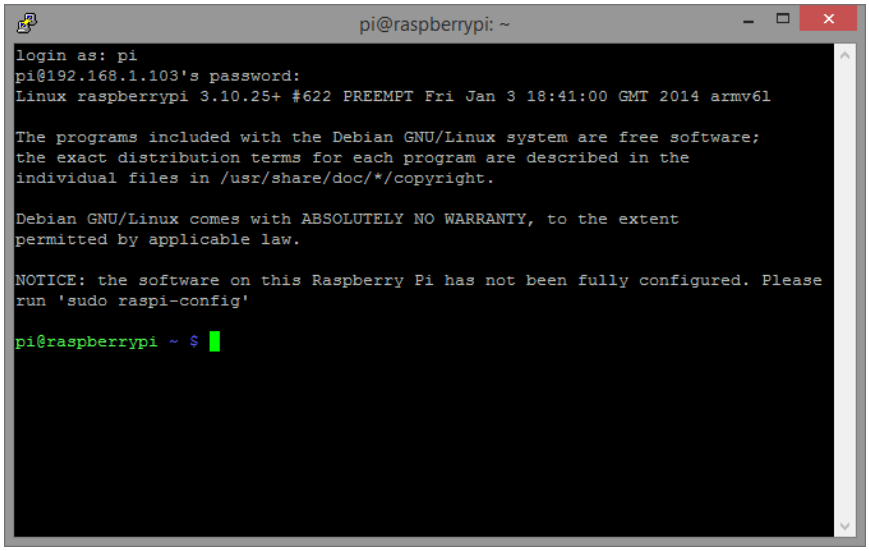


Fonte: O autor.

Ao estabelecer a conexão pela primeira vez (figura 10) foi utilizado o usuário “pi” e a senha “raspberry” para login no sistema. A partir daí foi alterado algumas configurações e atualizado o sistema com os respectivos comandos:

- *sudo raspi-config*
- *sudo apt-get update*
- *sudo apt-get upgrade*

Figura 10 - Conexão como *Raspbery Pi* através da ferramenta *Putty*



Fonte: O autor

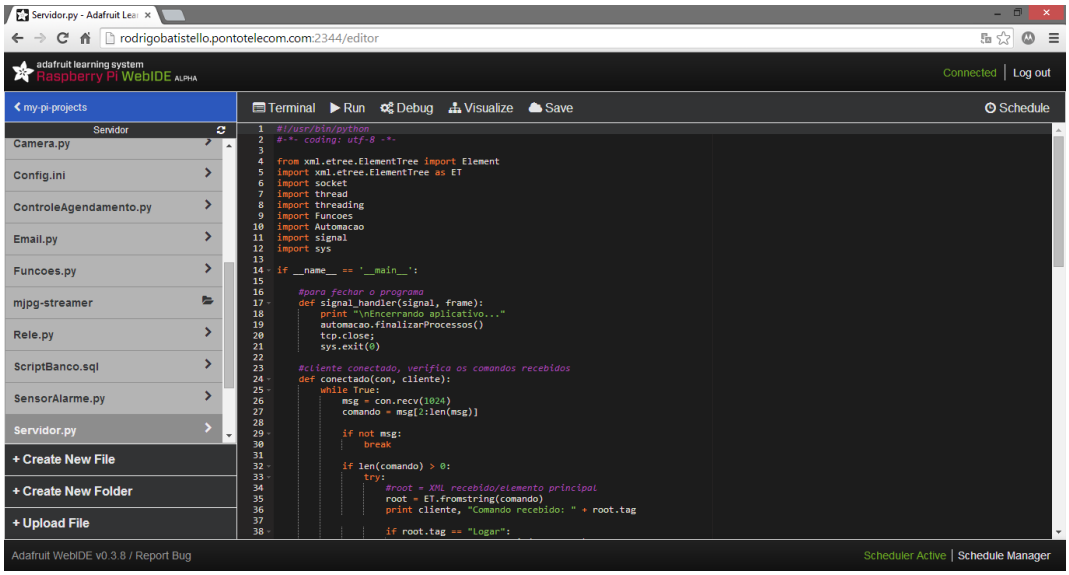
5.2.2 Instalação da IDE de desenvolvimento no *Raspberry Pi*

Para desenvolver o trabalho na linguagem Python no *Raspberry Pi* optou-se pela utilização da *Integrated Development Environment* (IDE) desenvolvida pela Adafruit, denominada WebIDE, a empresa desenvolvedora é a mesma que desenvolveu algumas das bibliotecas utilizadas neste trabalho. Esta IDE facilita muito o desenvolvimento, uma vez que possuí uma interface web, simples e muito eficiente. Segundo Cooper (2012) a instalação da mesma é possível através da execução do seguinte comando: *curl https://raw.githubusercontent.com/adafruit/Adafruit-WebIDE/alpha/scripts/install.sh | sudo sh.*

Realizado o procedimento de instalação foi alterado a porta de acesso por meio de um navegador, acessando um endereço no seguinte padrão: http://IP do Raspberry Pi/config (COOPER, 2012).

Depois de alterar a porta foi necessário reiniciar o *Raspberry Pi*. Feito isso bastou apenas acessar novamente o endereço removendo o *config* do final (http://IP do Raspberry Pi). Nesta etapa foi necessário configurar uma conta no *Bitbucket* que nada mais é que um repositório de fontes online, privado e gratuito. Nesta etapa foi exibido na tela todos os passos que devem ser seguidos para obter acesso a IDE que pode ser observada na figura 11. (COOPER, 2012).

Figura 11 - Interface da IDE para desenvolvimento em Python



Fonte: O autor.

5.2.3 Configuração dos pinos GPIO

Para utilização dos pinos GPIO que permitem o controle do sistema de automação, foi necessário seguir alguns passos para configurar os mesmos, esses passos são disponibilizados pela empresa que desenvolve a biblioteca e podem ser facilmente executados no sistema.

Seguindo as orientações de Monk (2012) foi necessária instalação de dois pacotes obtidos pelos comandos:

- *sudo apt-get install python-dev*
- *sudo apt-get install python-rpi.gpio*

Depois foi editado um arquivo com o seguinte comando:

- *sudo nano /etc/modules*

E adicionado as seguintes linhas:

- *i2c-bcm2708*
- *i2c-dev*

Instalado mais dois pacotes:

- *sudo apt-get install python-smbus*
- *sudo apt-get install i2c-tools*

Editado o arquivo e removida as entradas da *blacklist* comentando as linhas (#):

- *sudo nano /etc/modprobe.d/raspi-blacklist.conf*
- *# blacklist spi-bcm2708*
- *# blacklist i2c-bcm2708*

5.2.4 Instalação do banco de dados

Para armazenar as informações necessárias no sistema de automação foi optado pelo uso do banco de dados MySQL junto ao aplicativo servidor. Essa opção foi devido a facilidade de instalação, utilização, integração com a linguagem Python e também pelo mesmo ser relativamente leve, funcionando sem problemas no *Raspberry Pi*.

A instalação foi simples obtida através da execução de um único comando no terminal do Linux, foi necessário também configurar a senha de acesso para o usuário *root* do banco de dados quando solicitado (RASPBERRY WEB SERVER, [2013?]).

- *sudo apt-get install mysql-server python-mysqldb*

5.2.5 Instalação do gerenciador do banco de dados

Para gerenciar o banco de dados foi utilizado o aplicativo *phpMyAdmin*. Essa ferramenta é bastante útil pois facilita a criação e manipulação dos objetos do banco de dados. Ela roda em um servidor web, portanto pode ser acessada facilmente através de um navegador.

Seguindo a orientação de Fulton (2013) a instalação foi bem simples bastando a execução do comando abaixo e informando os dados de usuário e senha do banco quando solicitado.

- *sudo apt-get install phpmyadmin*

5.2.6 Instalação do servidor FTP

Também foi instalado no *Raspberry Pi* um servidor *File Transfer Protocol* (FTP) para possibilitar a transferência de arquivos para o mesmo. Esses arquivos consistem principalmente nas músicas que são executadas a partir do aplicativo.

Para efetuar a instalação foi seguido as orientações de Pinto (2013) e executado os seguintes comandos:

- *sudo apt-get install vsftpd*
- *sudo nano /etc/vsftpd.conf*

  E editar as seguintes opções:

  Desativar o acesso “anónimo”
- *anonymouse_enable=NO*

  Permitir o acesso dos utilizadores locais (criados no Linux)
- *local_enable=YES*

- *write_enable=YES*

  Alterar Banner do serviço
- *ftpd_banner*=Bem-Vindo ao FTP House Pi

  Por fim, salvar o arquivo e reiniciar o serviço.
- *sudo /etc/init.d/vsftpd restart*

5.2.7 Instalação do servidor de câmera

Para s*treaming* de vídeo da câmera conectada ao *Raspberry Pi* foi utilizada a biblioteca *mjpg-streamer*. Segundo IQ JAR (2013) a instalação da mesma consiste na execução das seguintes etapas:

- *sudo apt-get install libv41-0*
- *wget http://iqjar.com/download/jar/soft/mjpg-streamer.tar.gz*
- *tar -zxvf mjpg-streamer-rpi.tar.gz*
- *cd mjpg-streamer*
- *./mjpg-streamer.sh start*

Onde: mjpg-streamer.sh [start|stop|restart|status] [porta] [resolução] [frames]. Exemplo para visualização da imagem: http://raspberrypi:8083/?action=stream

5.2.8 Instalação do MPlayer

Para execução das músicas através do aplicativo foi utilizado um player muito conhecido no Linux o MPlayer. A instalação do mesmo é realizada com a execução do seguinte comando:

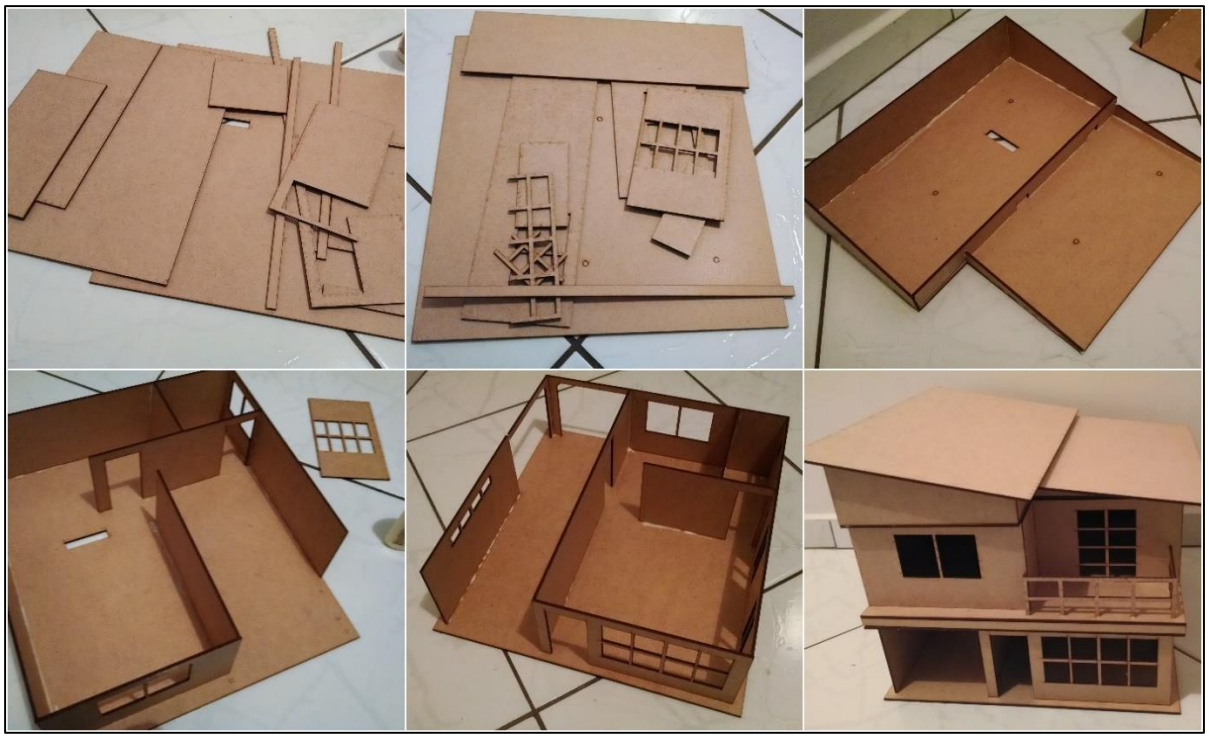
- *sudo apt-get install mplayer*

5.3 CONSTRUÇÃO DO PROTÓTIPO

A construção do protótipo foi de extrema importância neste trabalho, através dele foi possível ver de fato como se comporta uma residência automatizada. Os componentes utilizados foram adquiridos em lojas de componentes eletrônicos, não optou-se por uma marca específica foi apenas utilizado o que estava disponível. A ideia foi montar uma casa onde fosse possível controlar a iluminação (representada através de leds), um sistema de alarme, uma câmera, um sensor de temperatura e humidade, além de uma tomada para ligar algum outro tipo de equipamento.

Esse protótipo foi construído com uma maquete de *Medium Density Fiberboard* (MDF) adquirida através do site Mercado Livre, a mesma vem desmontada sendo necessário colar suas partes. O processo de montagem pode ser observado na figura 12.

Figura 12 - Montagem da Maquete



Fonte: O autor.

Finalizada esta etapa foi envernizada a madeira para maior durabilidade, e fixada a mesma sobre uma base para representar o jardim da residência. O protótipo pronto pode ser visualizado na figura 13.

Figura 13 - Protótipo para demonstração do sistema de automação

Fonte: O autor.

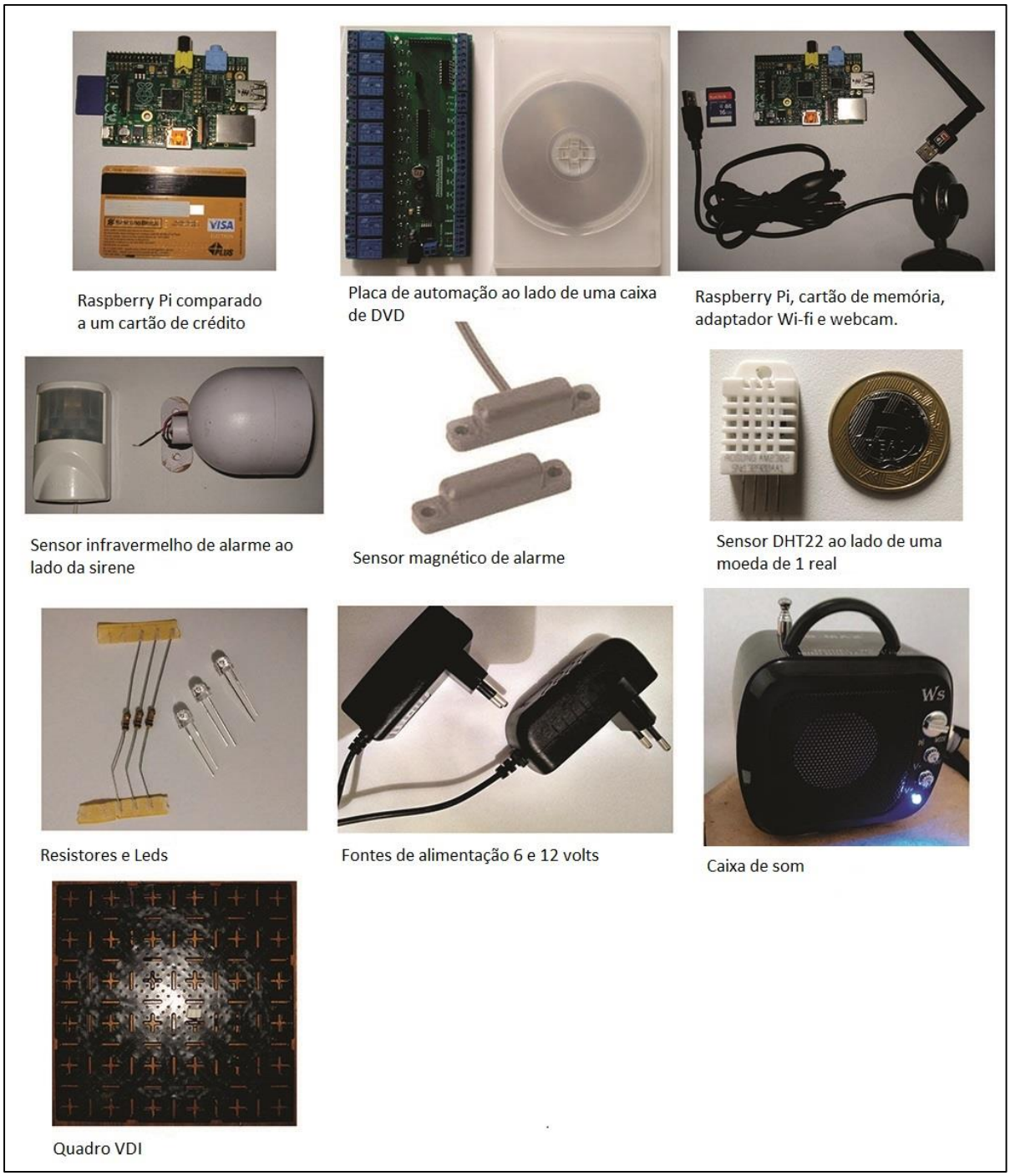
Depois disso foi construído a estrutura elétrica do sistema, iniciando pelo painel de controle que foi montado sobre uma base de Voz, Dados e Imagem (VDI), esta base permite a fixação dos componentes e a organização dos cabos, esse painel fica localizado no interior da maquete, mais precisamente no sobrado da casa. Na estrutura elétrica foi utilizado:

- 1 Quadro VDI;
- 1 *Raspberry Pi* modelo B;
- 1 Placa de automação criada pelo Projeto Arduino;
- 1 Sensor DHT22 (Sensor de temperatura e humidade);
- 1 Sirene de alarme comum 12 volts;
- 1 Adaptador *Wi-Fi*;
- 1 Cartão de memória SD de 16GB;
- 1 Sensor de alarme infravermelho;
- 1 Sensor de alarme magnético;
- 1 *Webcam* USB;
- 8 Leds verde *straw hat* (3.6 volts);
- 9 Leds branco frio *straw hat* (3.6 volts);
- 18 Resistores 10k (Para ligar os leds em 6 volts e o sensor DHT em 5 volts);
- 1 Fonte 12 volts e 2 amperes (alimentação das placas);
- 1 Fonte 6 volts e 2 amperes (alimentação dos leds);
- 1 Caixa de som;

- Fios, conectores, parafusos, presilhas, fita isolante e dupla face;

Os componentes utilizados podem ser observados na figura 14, alguns deles foram comparados a outros objetos para melhor noção de tamanho. Nem todos os componentes estão na mesma escala da maquete, por isso é necessário adaptá-los, como no caso da sirene que foi desmontada posteriormente para caber no protótipo.

Figura 14 – Componentes utilizados para montagem do sistema de automação

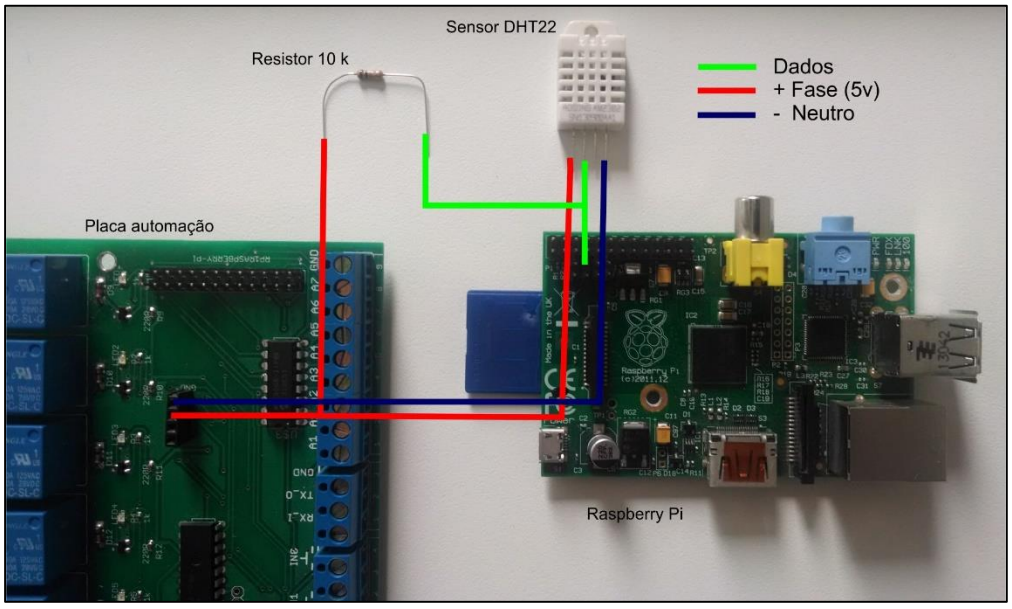


Fonte: O autor.

A conexão entre a placa de automação e o *Raspberry Pi* é feita através de um cabo flat, basta apenas prestar atenção no lado correto de conexão, identificado pela tarja branca/vermelha do cabo.

Já a conexão com os demais sensores e equipamentos é um pouco mais complexa. No caso do sensor de temperatura e humidade a conexão foi feita com o pino GPIO 28 do *Raspberry Pi*. Para que essa conexão funcione é necessário ligar o sensor em uma fonte de energia de 5 volts e também a um resistor de 10k, essa ligação é representada pela figura 15.

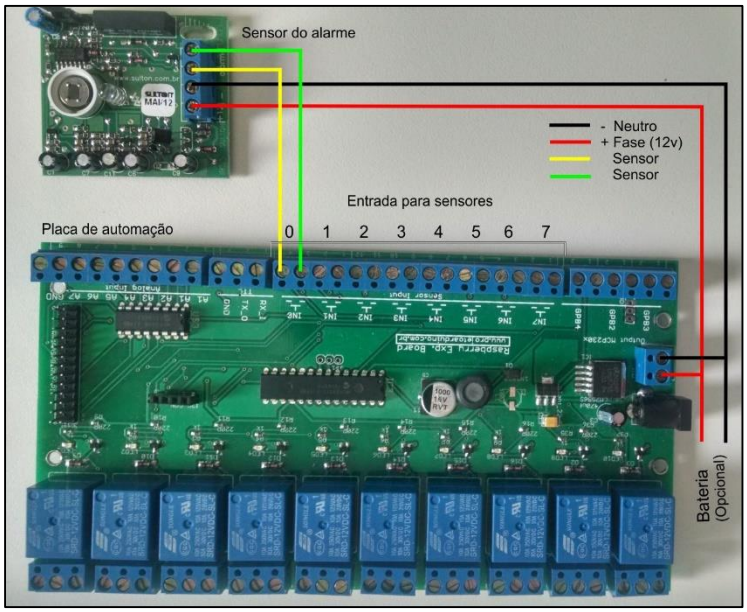
Figura 15 - Esquema elétrico do sensor de temperatura e humidade



Fonte: O autor.

Para ligar os sensores de alarme bastou conecta-los na placa de automação, no caso de sensores infravermelhos (destinados a detecção de movimento no ambiente) é necessário conectar também a uma fonte de alimentação, no caso do sensor utilizado 12 volts, para o sensor magnético (destinado a portas e janelas, que é violado sempre que uma das partes afasta-se da outra) não é necessária uma fonte de alimentação. O esquema de ligação pode ser observado na figura 16 onde é utilizado um sensor infravermelho comum, facilmente encontrado em lojas de componentes eletrônicos.

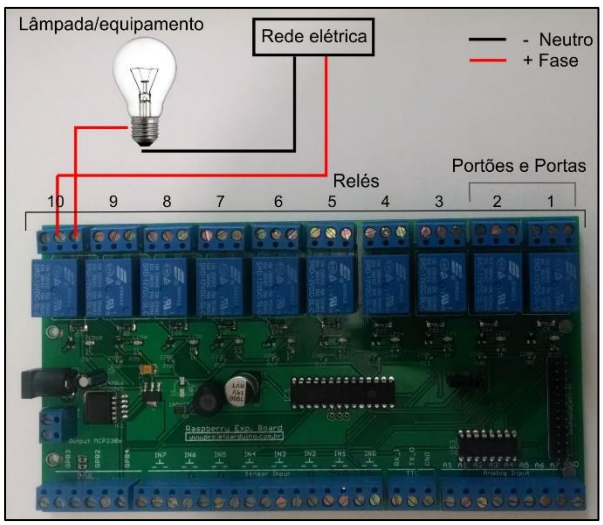
Figura 16 - Esquema de conexão dos sensores de alarme



Fonte: O autor.

O esquema de conexão dos relés pode ser observado na figura 17. Para funcionar é necessário conectar o cabo positivo (preferencialmente) da rede elétrica na entrada do meio do relé, este funciona como um interruptor, observando o lado esquerdo do relé temos uma saída que está sempre ligada e pode ser utilizada para acionamento manual através de um interruptor convencional, e temos também a saída direita que foi utilizada neste protótipo que é acionada através do sistema proposto, esta deve ser ligada no equipamento junto com o outro polo da rede elétrica. A inversão dos polos (Neutro e Fase) não interfere no funcionamento.

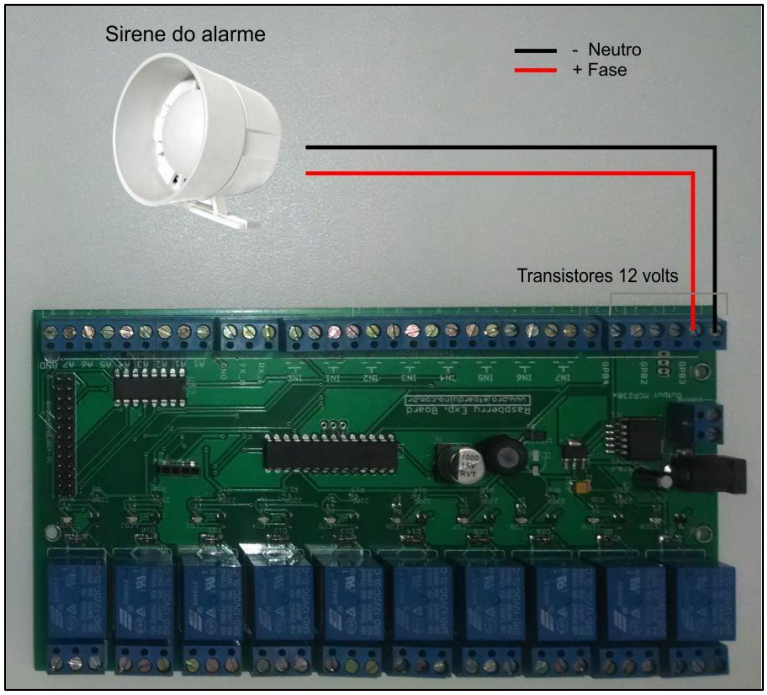
Figura 17 - Esquema elétrico dos relés



Fonte: O autor.

Para ligar a sirene do alarme foi utilizada uma das três saídas acionadas através de transistores da placa, essas saídas são de 12 volts e de no máximo 1 ampere. No caso foi optado pela saída acionada pelo GPIO 11, como pode ser observado na figura 18.

Figura 18 - Esquema de conexão da sirene

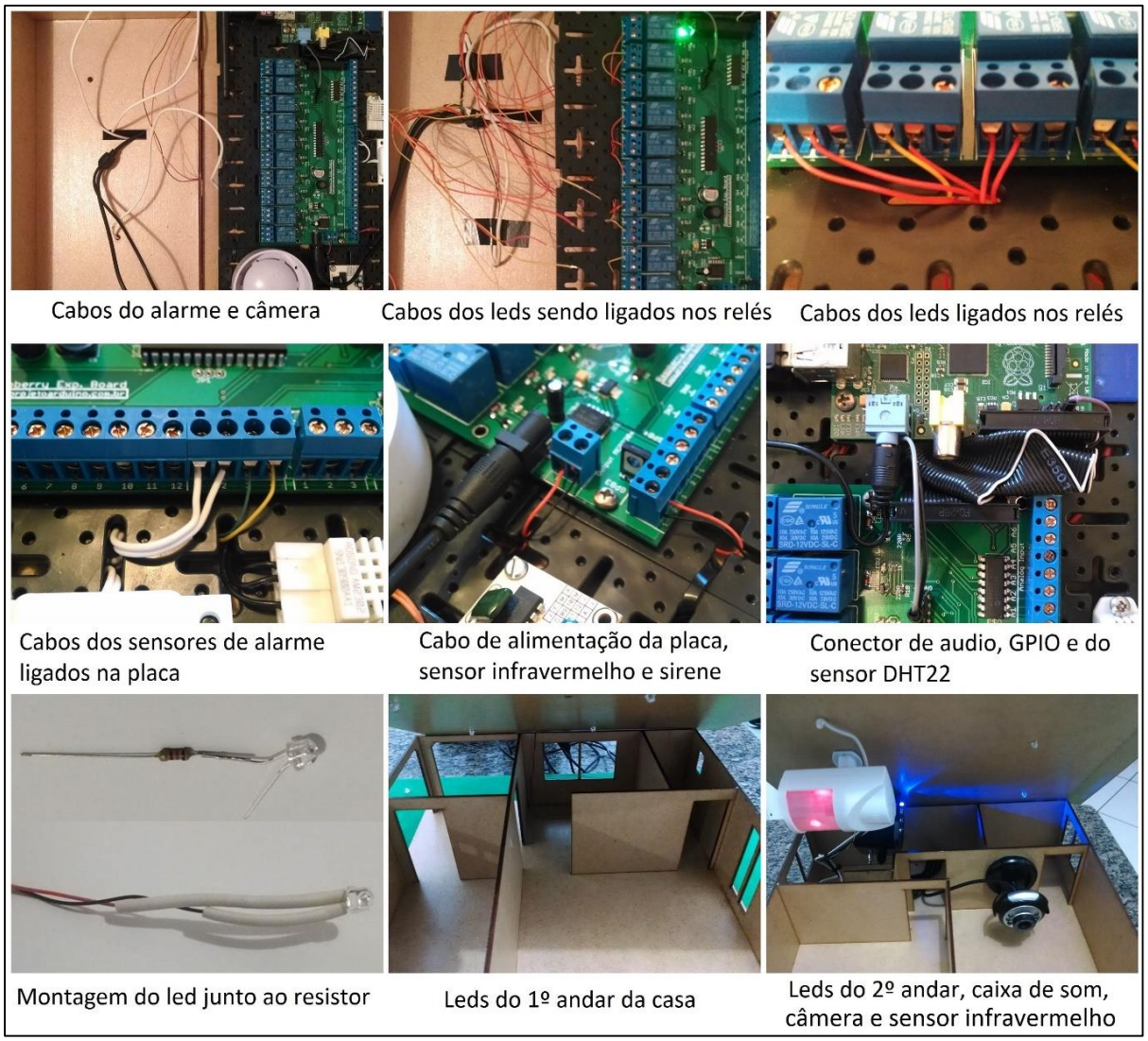


Fonte: O autor.

Depois de conhecer cada ligação elétrica foi iniciada a montagem do painel de controle, conectando todas as placas em cima do quadro VDI e ligando cada componente do protótipo. A figura 19 apresenta esse processo de montagem enquanto a figura 20 apresenta o painel de controle já montado no protótipo.

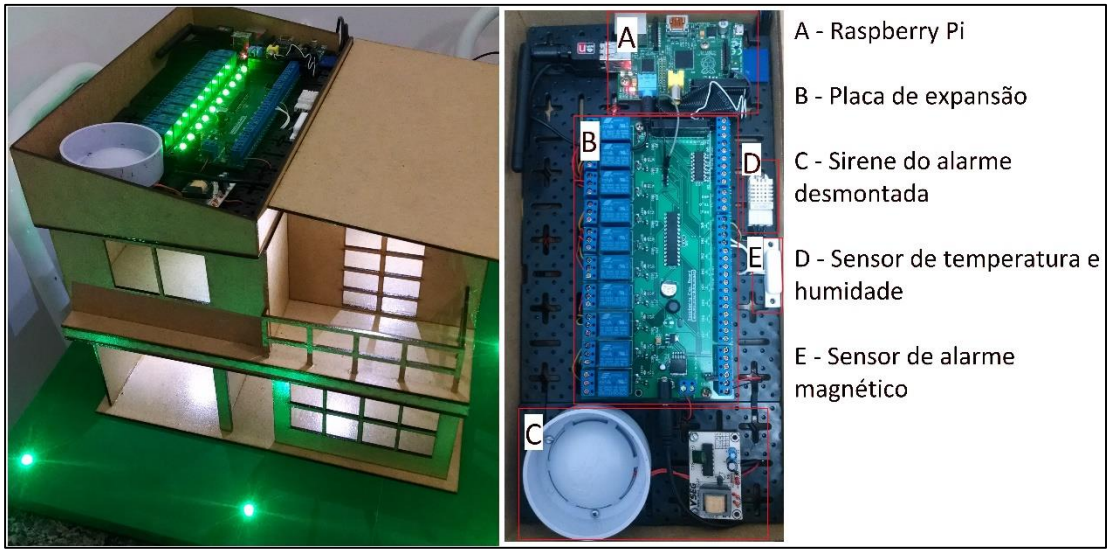
Como trata-se de um protótipo a alimentação dos leds utilizados é feita com uma tensão muito menor que a convencional em uma residência (110/220 volts). Isso é necessário devido a escala da maquete, pois não seria possível ligar lâmpadas convencionais na mesma. Porém estas ligações são as mesmas utilizadas em uma casa real. Para representar o acionamento de um equipamento de maior voltagem, foi utilizado um dos relés da placa como tomada, assim é possível acionar um equipamento de 220 volts por exemplo. Os acionamentos serão demonstrados mais adiante, no capítulo Funcionamento do Sistema.

Figura 19 - Processo de montagem da parte elétrica e do painel de controle



Fonte: O autor.

Figura 20 - Painel de controle do sistema de automação

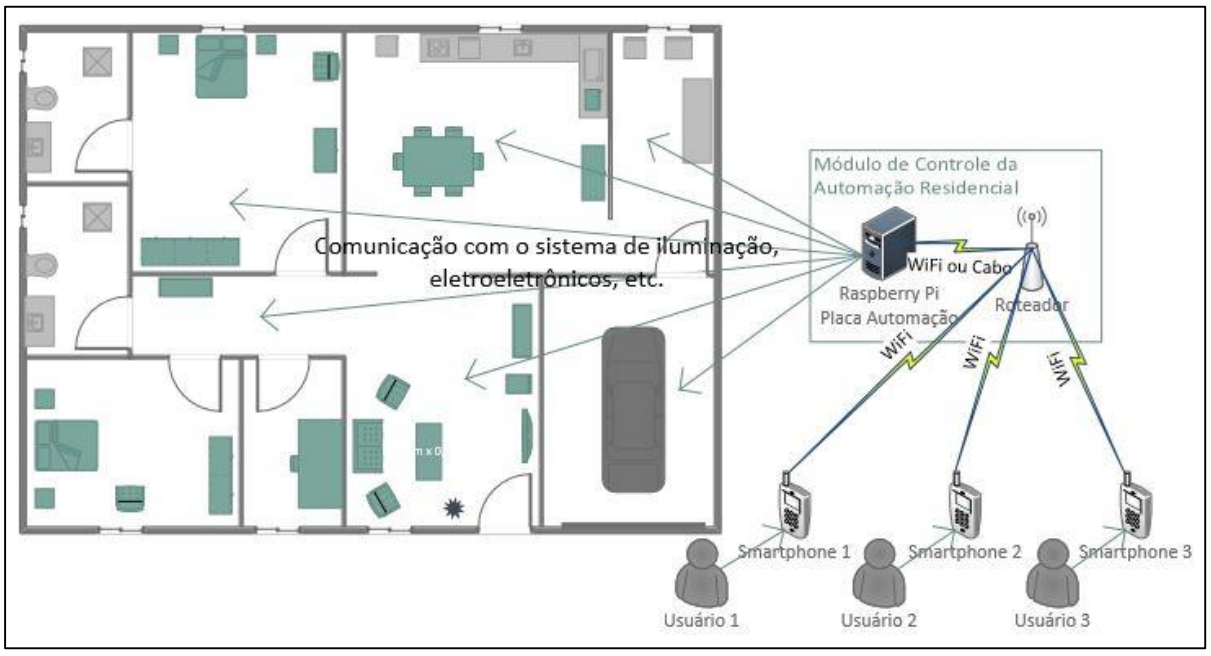


Fonte: O autor.

5.4 DESENVOLVIMENTO DO SISTEMA

O sistema consiste basicamente em um aplicativo móvel desenvolvido para plataforma Android na linguagem Java, o mesmo comunica-se através de uma rede *Wi-Fi* ou 3G com o *Raspberry Pi* por intermédio de um roteador. No *Raspberry Pi* tem o servidor desenvolvido em Python, que controla o sistema de automação. O mesmo fica constantemente aguardando conexões, assim que uma conexão for estabelecida e um comando for enviado o servidor interpreta o mesmo e repassa para a placa de automação que por sua vez aciona o equipamento ou sensor desejado, o sucesso ou não do comando é informado ao dispositivo que o requisitou, esse processo está ilustrado na figura 21.

Figura 21 - Comunicação entre os dispositivos móveis e o sistema de automação



Fonte: O autor.

5.4.1 O aplicativo servidor

Como já mencionado anteriormente o aplicativo servidor foi desenvolvido em Python, o mesmo utiliza como banco de dados o MySQL, para criação do banco de

dados foi utilizada a ferramenta phpMyAdmin, o script das tabelas foi gerado pela ferramenta utilizada para construção do MER, no caso o Case Studio, porém foi necessário a inserção de alguns registros de forma fixa, como os relés e os sensores de alarma da placa. A IDE de desenvolvimento utilizada foi a WebIDE da Adafruit.

Para seu desenvolvimento foram utilizadas duas bibliotecas externas, estas são responsáveis pelo controle dos pinos GPIO: Adafruit_MCP230xx (instalada junto com a IDE) e a RPi.GPIO (instalada ao configurar os pinos GPIO). Ambas as bibliotecas são disponibilizadas de forma gratuita por seus desenvolvedores.

Este aplicativo é iniciado automaticamente sempre que o *Raspberry Pi* é ligado, isso foi feito adicionando a linha abaixo no arquivo */etc/init.d/rc.local* do sistema operacional.

- *nohup sudo python /usr/share/adafruit/webide/repositories/my-pi-projects/Servidor/Servidor.py& </dev/null >/dev/null 2>&1 &*

Este comando é responsável por iniciar o aplicativo servidor a partir do código fonte principal do mesmo, de forma com que ele não fique dependente de um terminal.

5.4.2 O Aplicativo cliente

O aplicativo cliente foi desenvolvido para a plataforma Android utilizando o *Android Developer Tools* (ADT), que nada mais é que um conjunto de ferramentas integradas a IDE de desenvolvimento Eclipse, muito conhecida e utilizada para desenvolvimento em Java.

O aplicativo desenvolvido é compatível com a versão 2.3.6 do Android ou superior, para isto foi utilizado as bibliotecas de compatibilidade do Android: *v4 Support Library* e *v7 Libraries*. Também foi utilizada a biblioteca jdom-2.0.5 (http://www.jdom.org/downloads/index.html) para manipulação dos arquivos *Extensible Markup Language* (XML), *MjpegView* (https://code.google.com/p/android-camera-axis/) para visualização da imagem da câmera e o *DateTimePickerDialog* (https://github.com/nguyentoantuit/android/tree/master/DateTimePicker) para preenchimento de informações de data e hora. Para que o visual do aplicativo se mantivesse igual em todas as versões do Android foi utilizado o gerador de temas

Android Holo Colors Generator (http://android-holo-colors.com), todas essas bibliotecas são disponibilizadas gratuitamente por seus desenvolvedores.

O aplicativo recebeu o nome de *House Pi*, sendo *House* de casa e *Pi* de *Raspberry Pi.* O ícone do aplicativo segue a ideia do nome, sendo uma casa e uma framboesa.

5.4.3 Funcionamento do sistema

Para que tudo funcionasse foi necessário realizar a integração entre as diferentes linguagens utilizadas e também entre o aplicativo servidor e o *hardware*. Para isso, foi necessário o estabelecido de padrões de comunicação.

O aplicativo desenvolvido em Python nada mais é que um servidor socket, ele fica constantemente escutando uma porta pré-configurada, essa configuração bem como outras necessárias ficam armazenadas no arquivo “conf.ini”. Quando uma conexão é recebida uma *thread* é levantada, permitindo assim que novas conexões sejam aceitas e múltiplos usuários manipulem o sistema.

Todos os clientes conectados compartilham a classe principal do sistema denominada Automacao e consequentemente as demais classes, assim é possível consultar os status atuais dos relés, alarme, entre outros.

Todas as mensagens enviadas para o servidor são no formato de XML, esse XML é montado no aplicativo cliente, transformado em texto e enviado ao servidor, este transforma o texto em XML novamente para posterior interpretação. O conteúdo transmitido foi codificado e decodificado em *8-bit Unicode Transformation Format* (UTF-8), para que os caracteres transmitidos permanecessem idênticos em ambos os lados, evitando-se assim problemas com caracteres especiais.

Tudo isso foi estabelecido para que ocorra um tratamento adequado que consiste em identificar o nó principal do XML, e através do seu nome direcionar ao método correto da classe, como pode ser observado no quadro 3. Como resposta, o aplicativo cliente poderá receber um texto que será transformado em XML, ou simplesmente uma identificação, como “Ok” ou “Erro”.

Assim é realizada a comunicação entre as linguagens, envia-se um comando e aguarda-se quando necessário uma resposta. As mensagens trocadas são em

formato de texto e determinados métodos transformam ou não estes textos em arquivos XML para posterior interpretação.

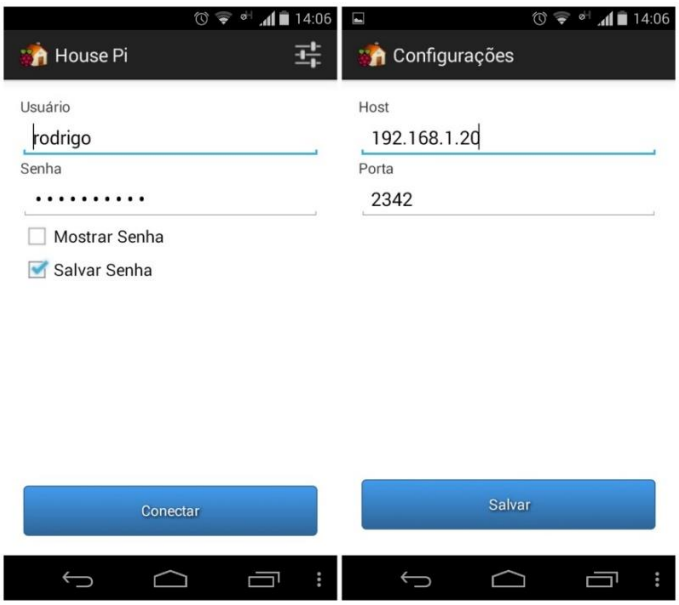
Quadro 3 – Trecho da rotina de interpretação do comando recebido (Servidor)

```
#cliente conectado, verifica os comandos recebidos
def conectado(con, cliente):
    while True:
        msg = con.recv(1024)
        comando = msg[2:len(msg)]

        if not msg:
            break

        if len(comando) > 0:
            try:
                #root = XML recebido/elemento principal
                root = ET.fromstring(comando)
                print cliente, "Comando recebido: " + root.tag

                if root.tag == "Logar":
                    automacao.efetuarLogin(root, con)
                elif root.tag == "Rele":
                    automacao.controlarRele(root, con)
                elif root.tag == "Temperatura":
                    automacao.enviarTemperaturaHumidade(con)
```

Fonte: O autor

A fim de conectar no servidor do sistema de automação a interface inicial do aplicativo cliente consiste em informar o login e a senha, além de um botão para conectar. O endereço de conexão pode ser alterado através da opção do menu configurações, como pode ser visualizado na figura 22.

Figura 22 – Interface inicial do sistema ao lado da configuração de acesso



Fonte: O autor.

Como o sistema não exige um registro de log por usuário existe apenas um usuário cadastrado no sistema, não é possível inserir novos usuário é possível apenas editar o existente.

No método correspondente ao estabelecimento da conexão com o servidor socket é instanciado duas variáveis, in e out, essas variáveis são utilizadas para o envio e o recebimento das mensagens através do código Java, como pode ser observado no trecho de código da classe Conexao no quadro 4.

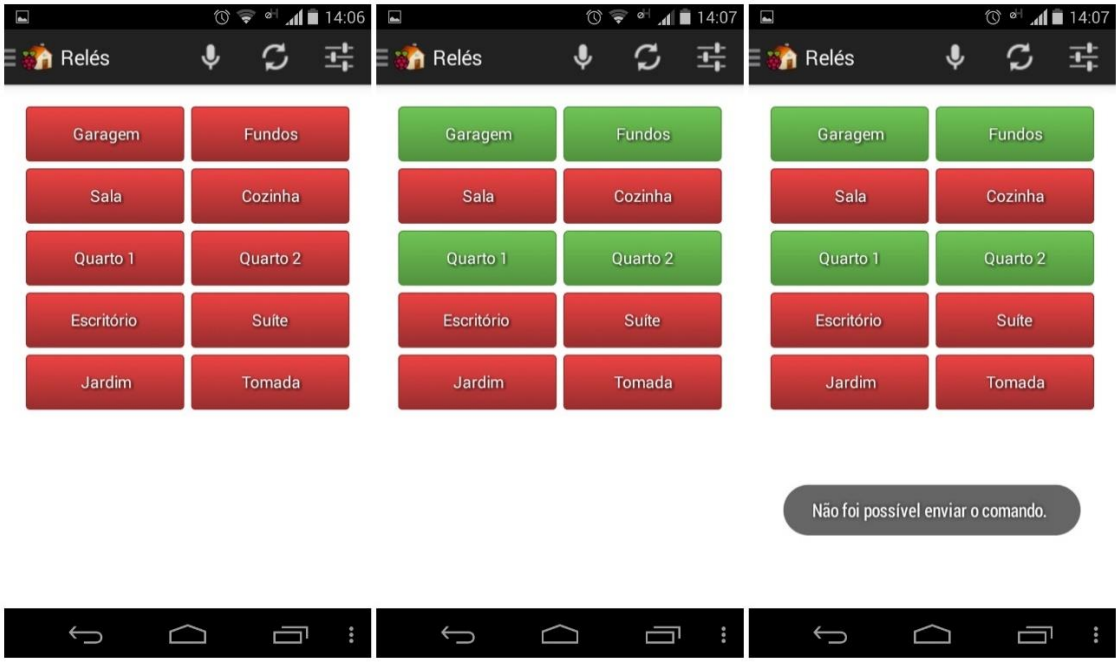
Quadro 4 - Trecho de código Java que estabelece a conexão com o servidor

```
public void conectar() throws Exception {
        InetSocketAddress inet = new InetSocketAddress(host, porta);
        String hostConexao = inet.getAddress().getHostAddress();

        this.socket = new Socket(hostConexao, porta);
        this.socket.setSoTimeout(15000);

        out = new DataOutputStream(socket.getOutputStream());
        in = new DataInputStream(socket.getInputStream());
}
```

Fonte: O autor.

Após efetuar o login no sistema é exibida a interface para controle dos relés, esta interface consiste em apresentar 10 botões que podem ser renomeados de acordo com o nome do equipamento conectado. O botão fica verde quando o equipamento está ligado e vermelho quando desligado. Para ligar ou desligar o equipamento basta clicar no botão, caso a operação seja efetuada com sucesso o botão altera a cor, do contrário será exibida a mensagem informando a falha na operação. Outra forma de ligar ou desligar o equipamento é pressionando o botão correspondente ao comando de voz e na sequência pronunciando o nome do equipamento. O comando de voz não funciona em todos os dispositivos, no caso da funcionalidade não estar disponível no aparelho será exibida uma mensagem informando ao usuário. Esta interface de controle pode ser observada na figura 23.

Figura 23 - Interface de controle dos relés (Aplicativo Cliente)



Fonte: O autor.

Ao clicar em ligar ou desligar um relé, um XML é montado para envio ao servidor, nele contém a informação do relé e da ação “Ligar” ou “Desligar”, chegando no servidor o XML é interpretado e direcionado ao método ControlarRele da classe Automacao, lá é identificado a operação e o relé, e então é chamado o método ligar ou desligar da classe Rele, depois de executada a operação é enviada a resposta ao cliente e atualizado o status no banco. O trecho de código em Python responsável por ligar ou desligar os relés pode ser visualizado no quadro 5.

Quadro 5 - Trecho de código responsável por ligar ou desligar o relé (Servidor)

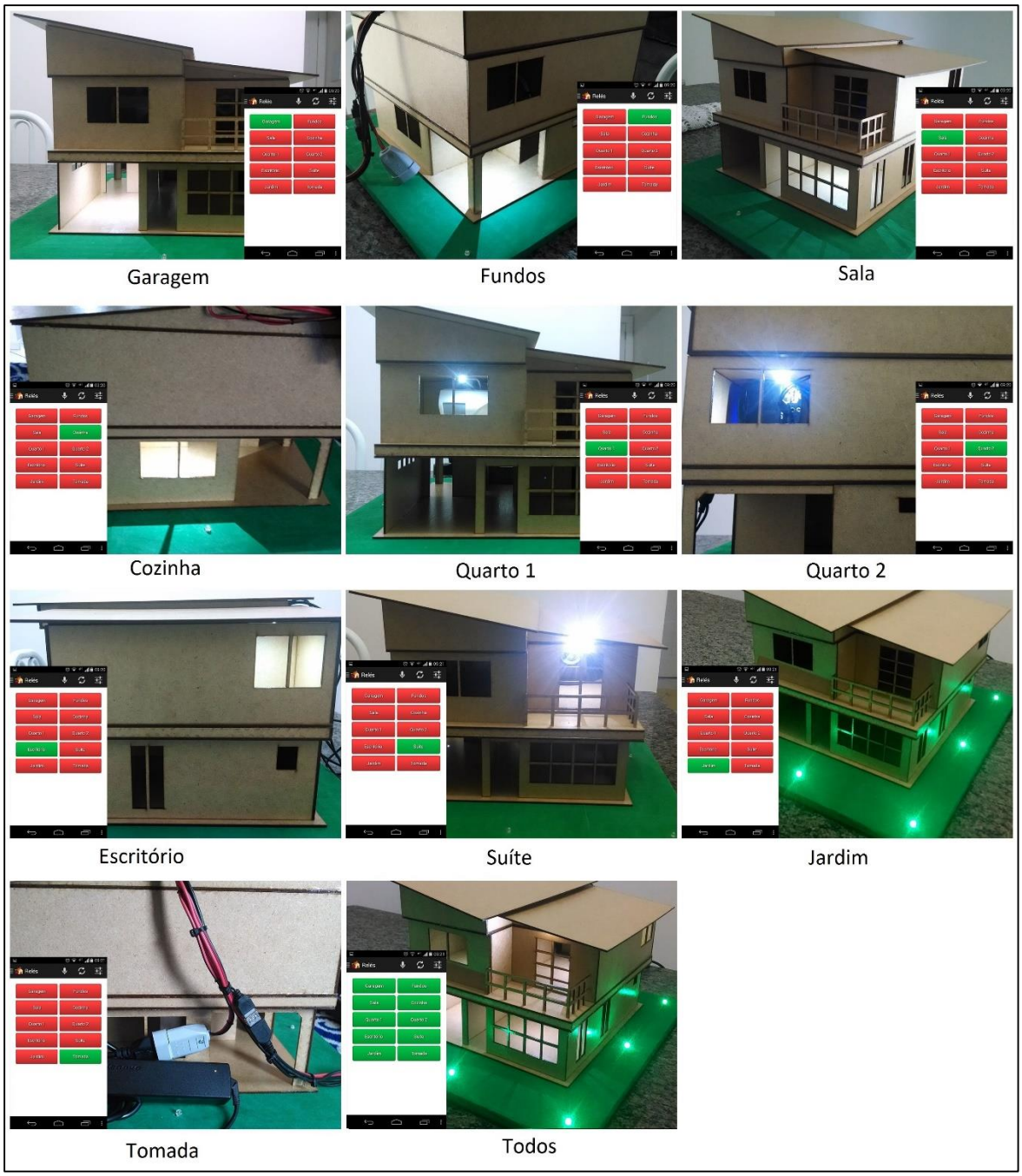
```
#funcao para ligar o rele
def ligar(self):
    try:
        mcp.output(self.numeroGPIO, LIGAR)
        self.status = STATUS_LIGADO
        return True
    except:
        return False

#funcao para desligar o rele
def desligar(self):
    try:
        mcp.output(self.numeroGPIO, DESLIGAR)
        self.status = STATUS_DESLIGADO
        return True
    except:
        return False
```

Fonte: O autor.

Na figura 24 pode ser observado passo a passo como fica o protótipo desenvolvido de acordo com o acionamento de cada relé, abaixo de cada imagem está escrito o nome do relé acionado no sistema. Na figura 25 é possível visualizar uma imagem ampliada do protótipo com todos os relés ligados.

O sistema também apresenta uma proteção contra falhas, essa proteção consiste em retornar ao estado anterior no caso de um desligamento inesperado. Isso é possível através da gravação de status dos equipamentos no banco de dados.

Figura 24 - Controle dos relés x Efeito no protótipo



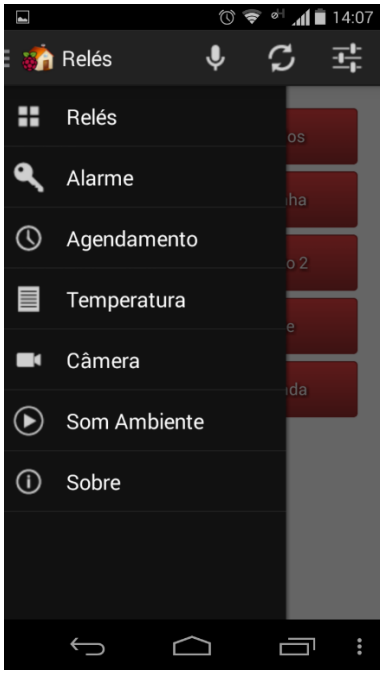
Fonte: O autor.

Figura 25 – Protótipo desenvolvido com todos os relés ligados

Fonte: O autor.

Para utilizar as outras funções do aplicativo desenvolvido é necessário acessar o menu lateral conhecido como *Navigation Drawer*. Uma forma de fazer isso é clicando no ícone do aplicativo. Este menu pode ser visualizado na figura 26.

Figura 26 - Menu de navegação do aplicativo para Android

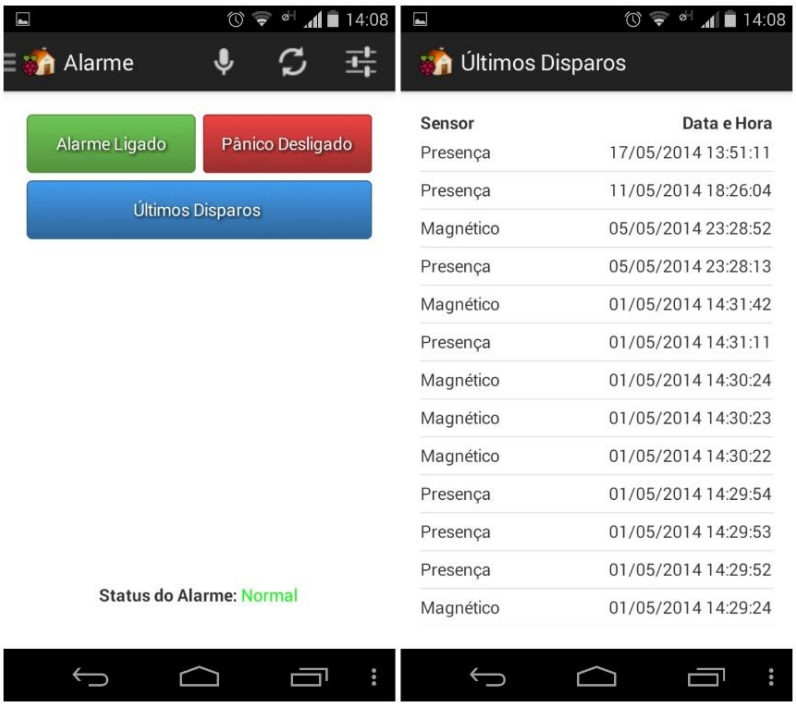


Fonte: O autor.

Acessando a opção de controle do Alarme é possível obter acesso há três botões, um deles liga e desliga o alarme, outro liga e desliga a função pânico

(acionamento imediato da sirene) e o outro dá acesso a uma interface onde é apresentado os últimos disparos. Essas interfaces podem ser visualizadas na figura 27.

Figura 27 - Interface de controle do alarme ao lado dos últimos disparos



Fonte: O autor.

O acionamento do alarme consiste na execução de um método como *thread* no servidor, isso é feito para que o usuário consiga continuar utilizando o aplicativo enquanto o mesmo está ativado. Este método está implementado na classe Alarme e consiste na leitura dos sensores ativos. Um sensor sem violação retorna o valor 1, enquanto um sensor violado 0. Toda vez que um sensor é violado é executado o procedimento de acordo com o que o usuário configurou, que pode ser: enviar um e-mail e disparar a sirene por 30 segundos, após isso volta-se a fazer a leitura dos sensores até que uma nova violação seja detectada ou o alarme desligado. O procedimento de leitura do sensor está implementado na classe SensorAlarme e pode ser visualizado no quadro 6.

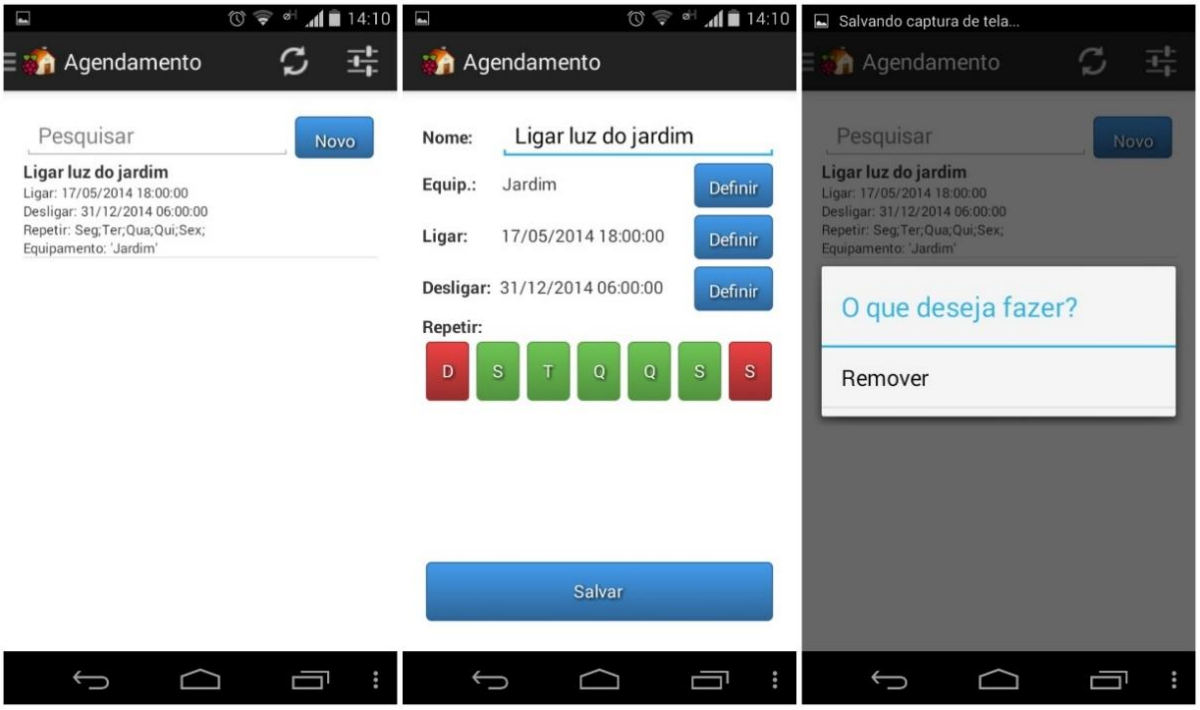
Quadro 6 - Trecho de código que faz a leitura do sensor de alarme (Servidor)

```
#funcao para ler o status do sensor (1 = normal e 0 = violado)
def lerStatus(self):
    return GPIO.input(self.numeroGPIO)
```

Fonte: O autor.

Outra função disponível no aplicativo é a criação de agendamentos, estes podem acionar um ou mais relés e/ou o alarme em um determinado dia e horário, além de poder repetir semanalmente ou diariamente até uma determinada data. A interface para visualização dos agendamentos cadastrados é simples, nela é possível obter acesso ao formulário de cadastro, campo para pesquisa ou remoção ao clicar sobre um agendamento e segurar pressionado. Essas interfaces podem ser visualizadas na figura 28.

Figura 28 - Interfaces de controle de agendamento

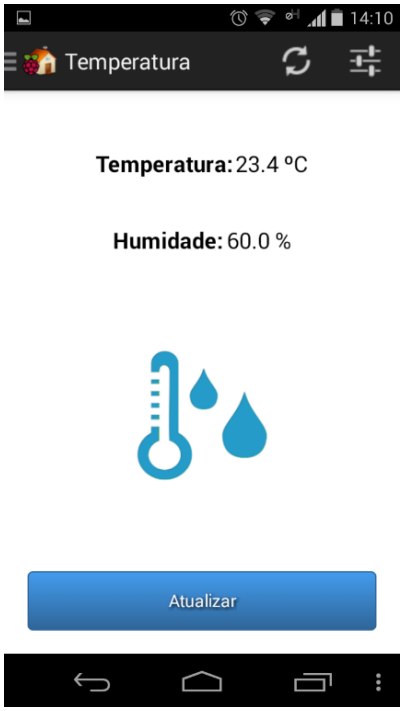


Fonte: O autor.

Os agendamentos cadastrados têm seus dados armazenado em três tabelas do banco de dados: Agendamento, DiaAgendamento e ReleAgendamento. Para controle no sistema existe uma lista com os agendamentos ativos, o processo de ligar ou desligar os equipamentos é feito através da *thread* implementada no fonte ControleAgendamento, está percorre a lista comparando a data e a hora atual com a do agendamento, quando forem iguais é executada a operação correspondente.

Existe também uma opção que dá acesso a temperatura e a humidade do ambiente, assim o usuário pode por exemplo tomar a decisão de ligar ou não um climatizador. A interface correspondente a esta funcionalidade pode ser visualizada na figura 29.

Figura 29 - Interface para visualização de temperatura e humidade



Fonte: O autor.

A obtenção dos dados de temperatura e humidade consiste na utilização do driver desenvolvido pela Adafruit, como este driver é escrito na linguagem C, o mesmo é manipulado pelo terminal do Linux através da classe *subprocess* do Python e a reposta é obtida através do que foi impresso no próprio terminal e enviado ao dispositivo que requisitou. A utilização do driver pode ser observado no quadro 7.

Quadro 7 - Trecho do método que obtém a temperatura e a humidade (Servidor)

```
class TemperaturaHumidade(object):

    def __lerSensor(self):
        return subprocess.check_output([Funcoes.lerConfiguracaoIni("CaminhoDHT"),
                                        Funcoes.lerConfiguracaoIni("TipoDHT"),
                                        Funcoes.lerConfiguracaoIni("GPIODHT")]);

    def getDados(self):
        output  = self.__lerSensor();
        matches = re.search("Temp =\s+([0-9.]+)", output)

        # [...] Tenta novamente se não conseguir

        temp = float(matches.group(1))
        matches = re.search("Hum =\s+([0-9.]+)", output)

        # [...] Tenta novamente se não conseguir

        hum = float(matches.group(1))
```

Fonte: O autor.

Outra função existente dá acesso a visualização da câmera, sua função é exibir a imagem transmitida através de *streaming* de uma câmera presente na residência e conectada ao *Raspberry Pi*. Tanto o envio quanto o recebimento são feitos através da biblioteca *mjpg-streamer*. Quando o aplicativo móvel solicita a visualização da câmera é disparado um comando que inicia o serviço de *streaming*, quando o mesmo não é mais necessário é disparado um comando que para a execução o serviço. Trechos de código desta funcionalidade podem ser observados no quadro 8 enquanto a interface do sistema pode ser visualizada na figura 30.

Quadro 8 - Trechos de código correspondentes ao controle da câmera

Trecho de código Java:

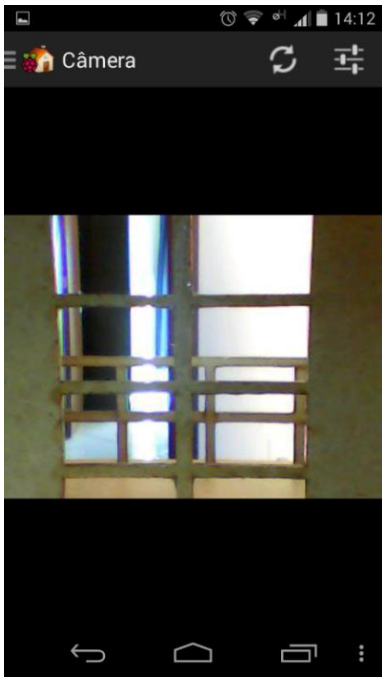
```
String url = "http://" + Login.IP_SERVIDOR + ":" +
             portaCamera.toString() + "/?action=stream";

mv = (MjpegView) rootView.findViewById(R.id.mjpeg_view);
connection(mv, url);
```

Trecho de código Python:

```
#inicia o servico da camera
def ligar(self):
    porta = int(Funcoes.lerConfiguracaoIni("Porta")) + 1
    os.system("sudo " + self.MJPG + " start " + str(porta) + " " +
              Funcoes.lerConfiguracaoIni("ConfiguracaoMJPG"))

#para o servico da camera
def desligar(self):
    os.system("sudo " + self.MJPG + " stop")
```

Fonte: O autor.

Figura 30 - Interface para visualização da imagem da câmera

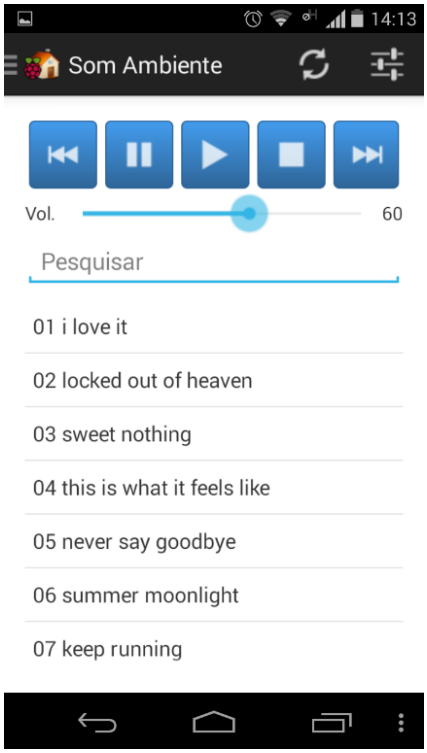


Fonte: O autor.

Durante a transmissão da imagem da câmera, o uso do processador pode ser alto, tanto no servidor quando no dispositivo móvel, isso gera alguns impactos. Um exemplo é a obtenção dos dados de temperatura e humidade no servidor, estes não são retornados enquanto o serviço de câmera está sendo executado, justamente pelo fato do processador estar ocupado com outra tarefa.

Outra implementação foi uma interface para controle de som ambiente, como pode ser visualizado na figura 31. Nesta interface é possível pesquisar a música, executar, pausar, avançar, retroceder ou parar. Sempre que executado uma destas operações é disparado um XML com o comando. No servidor o controle é feito através do MPlayer juntamente com um arquivo de PIPE. Toda operação é gravada neste arquivo e o MPlayer por sua vez lê, interpreta e executa o comando. A manipulação desse arquivo é feita através da classe *subprocess* do Python que permite a manipulação do terminal do Linux durante a execução do aplicativo.

Figura 31 - Interface de controle do som ambiente



Fonte: O autor.

As músicas para execução devem ser adicionadas na pasta pré-configurada no arquivo “conf.ini”. Isso deve ser feito através do servidor FTP instalado no *Raspberry Pi*. No quadro 9 é possível visualizar como é executada as músicas do diretório ao chamar a função “Play” da classe SomAmbiente.

Quadro 9 - Trecho de código Python que inicia a execução das músicas

```
cmd = ['mplayer', '-slave', '-quiet', '-playlist', self.__caminhoPlaylist]
self.__mplayer = subprocess.Popen(cmd, stdout=subprocess.PIPE, stdin=subprocess.PIPE)
```

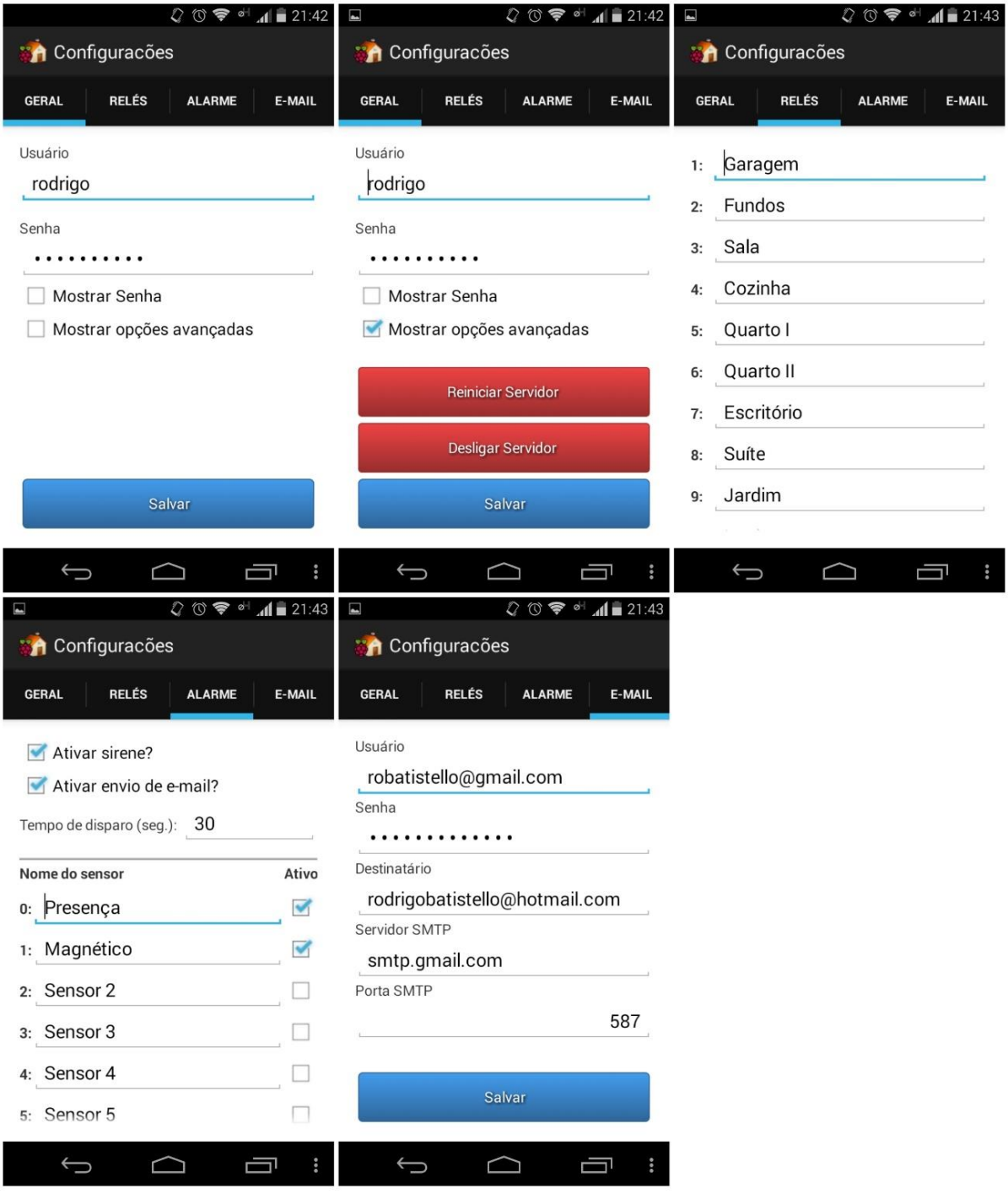
Fonte: O autor.

Além de toda parte de controle do sistema de automação residencial, existe também a parte de configuração (figura 32). Esta pode ser acessada a partir de qualquer interface de controle através do menu configurações. Essa parte de configuração está dividida em 4 abas: Geral, Relés, Alarme e E-mail. Todas as abas contam com um botão salvar no final que persiste as informações no banco de dados do servidor. Cada aba corresponde a modificação das seguintes configurações:

- Geral: É possível alterar o usuário e senha de acesso ao sistema de automação, além de visualizar as opções avançadas que consistem em desligar ou reiniciar o *Raspberry Pi*, útil já que não há outro meio de fazer isso a não ser utilizando um computador para conectar no mesmo e executar a operação.
- Relés: Local onde é possível alterar o nome que é exibido no botão de controle do relé, Ex.: Luz cozinha, Cafeteira.
- Alarme: Configura as opções de funcionamento do alarme, ativa ou não a sirene e o envio do e-mail, além de configurar o tempo que o alarme deve permanecer disparado. Nesta interface também é possível selecionar quais sensores estão ativos e o nome deles para facilitar a identificação na tela de visualização de disparos e no e-mail enviado.
- E-mail: Local onde é configurado quem deve receber o e-mail no caso de um disparo do alarme, além dos dados do servidor que será utilizado para envio do mesmo. Ex.: Gmail ou Hotmail. Esses dados consistem no servidor SMTP, porta, usuário e senha.

Ao alterar e salvar alguma configuração do sistema, são enviadas as novas configurações a classe correspondente no servidor e persistidas as mesmas no banco de dados, assumindo assim os novos parâmetros configurados.

Figura 32 - Interfaces de configuração do sistema



Fonte: O autor.

6 CONSIDERAÇÕES FINAIS

No decorrer do desenvolvimento deste trabalho foram levantadas todas as informações necessárias para o entendimento e a construção de um sistema de automação residencial. Os objetivos levantados foram todos alcançados com êxito, inclusive foram adicionadas ao sistema mais funcionalidades do que as inicialmente previstas.

O uso do *Raspberry Pi* em sistemas de automação é perfeitamente viável e possível, pode-se desenvolver praticamente tudo que sua criatividade permitir. É verdade que seu poder de processamento é baixo e algumas tarefas podem levar mais tempo para execução do que em um computador comum, mas no geral todas as funcionalidades do sistema de automação desenvolvido foram executadas sem problemas, o que torna seu uso ideal para sistemas deste tipo.

Optou-se pelo uso do *Raspberry Pi* e não de outra tecnologia como o Arduino, justamente por ele apresentar essa liberdade de desenvolvimento. Por ele ser um computador com um sistema operacional convencional você não fica preso a um sistema embarcado. Suas conexões GPIO, USB, rede *ethernet* e áudio/vídeo já integrados permitem maior facilidade de comunicação e possibilidades de desenvolvimento, apresentando assim todas as características necessárias para que ele seja um bom servidor de automação residencial.

A placa de automação utilizada é excelente, pois permite a execução de todas as tarefas a partir da linguagem de programação adotada e sem maiores complicações. Os demais componentes utilizados também não deixaram a desejar, não optou-se por marcas, apenas foram adquiridos componente comuns, vendidos em lojas de componentes eletrônicos, todos eles funcionaram sem problemas no protótipo, o que torna o custo benefício ainda melhor.

No que diz respeito as linguagens de programação adotadas pode-se dizer que adaptaram-se com mérito ao trabalho. Inicialmente houve uma dificuldade maior no desenvolvimento devido à própria falta de conhecimento na sintaxe das linguagens, principalmente no Python, porém com o passar do tempo o aprendizado foi evoluindo e o desenvolvimento do sistema passou a ser mais fácil. A troca de informações entre os aplicativos teve algumas complicações iniciais com caracteres especiais, porém os

problemas foram contornados fazendo a codificação e a decodificação em um determinado padrão.

Pode-se perceber que a lentidão principal no sistema passou a ocorrer em funções que exigiam manipulação do banco de dados, por isso foi otimizado o sistema para um melhor funcionamento e criado um MER compatível com o desempenho do *hardware*.

Na montagem do protótipo não houve maiores complicações, foi apenas necessário prestar atenção nas ligações elétricas para não queimar os componentes utilizados, todas as informações necessárias foram encontradas em materiais disponibilizados pelos fabricantes dos componentes e também na internet.

O principal objetivo deste trabalho era desenvolver um sistema capaz de controlar remotamente luzes, equipamentos eletroeletrônicos e um sistema de alarme inclusive realizando agendamentos para acionar os mesmos. Depois de alcançar esses objetivos foram exploradas e desenvolvidas outras funcionalidades como a leitura de um sensor de temperatura e humidade, transmissão via *streaming* da imagem de uma câmera e controle de um sistema de som ambiente. Tudo isso permitiu concluir que as possibilidades de uso do *Raspberry Pi* aliado a tecnologia móvel são infinitas.

A utilização dessas tecnologias ainda pouco exploradas como alternativa a sistemas de automação caros e pouco populares entre as pessoas se mostrou surpreendente, é possível tornar mais popular a automação residencial utilizando essas tecnologias, as mesmas podem facilitar muito a vida das pessoas, pois podem ser facilmente adaptadas com a realidade do ambiente e com um ótimo custo x benefício.

6.1 TRABALHOS FUTUROS

Como sugestão para trabalhos futuros pode ser ampliada as funcionalidades do sistema desenvolvido. No controle do alarme pode ser criada as chamadas zonas ou setores, que são áreas controladas juntas ou separadamente pelo sistema, assim seria possível ligar o alarme em apenas em algumas áreas da residência em determinados horários.

Também poderia ser criada uma opção que acionasse determinado relé ao disparar o alarme, como por exemplo as luzes externas da residência. Este disparo também poderia ser enviado ao dispositivo móvel na forma de notificação, isso seria uma segurança a mais caso o servidor de e-mail utilizado não estivesse funcionando no momento.

Poderia ser utilizado a leitura da temperatura e humidade para controlar um sistema de climatização, e assim ligar/desligar o mesmo quando a temperatura ambiente atingir o limite configurado.

Na parte de som ambiente poderia ser implementada uma rotina para enviar as músicas do dispositivo móvel ao sistema, dispensando assim o uso do FTP. Também poderia ser adicionado opções para reprodução de rádios on-line e informações sobre a faixa sendo executada no momento, bem como a criação até mesmo de um media center, utilizando a saída HDMI do *Raspberry Pi.*

Também poderia ser estudada uma maneira de expandir os relés da placa, permitindo o controle de um número maior de equipamentos. Outra expansão interessante é permitir a visualização de um número maior de câmeras de segurança.

REFERÊNCIAS

BOLZANI, Caio Augustus Morais. **Residências inteligentes:** Domótica, Redes Domésticas, Automação Residencial. 1 ed. São Paulo: Editora Livraria da Física, 2004. 332 p.

BORGES, Luiz Eduardo. **Python para Desenvolvedores**. 2. ed. Rio de Janeiro: Edição do Autor, 2010.

CADENHEAD, Rogers; LEMAY, Laura. **Aprenda em 21 dias:** Java 2. 4 ed. Rio de Janeiro: Elsevier, 2005.

CAMPOS, Augusto. **O que é Linux.** BR-Linux. Florianópolis, março de 2006. Disponível em <http://br-linux.org/faq-linux>. Acesso em: 01 set. 2013.

CESTA, André Augusto. **Tutorial**: A Linguagem de Programação Java e Orientação a Objetos. 2009. Disponível em: <

COOPER, Tyler. **Adafruit WebIDE:** Installation on Raspberry Pi. 2012. Disponível em: <https://learn.adafruit.com/webide/installation>. Acesso em: 15 abr. 2014.

FULTON, Bruce. **Install PHPMyAdmin on the Raspberry Pi.** 2013. Disponível em: <Install PHPMyAdmin on the Raspberry Pi>. Acesso em: 15 abr. 2014.

GUEDES, Gilleanes T.A. **UML 2**: Guia Prático. São Paulo: Novatec, 2007. http://www.ic.unicamp.br/~cmrubira/JAVATUT14PDF.pdf>. Acesso em: 29 set. 2013.

INTEL, Next Generation Center. **Mobilidade.** [200-?]. Disponível em: <http://www.nextgenerationcenter.com/detalle-curso/Mobilidade.aspx>. Acesso em: 07 out. 2013.

IQ JAR. **Live webcam feed from your Raspberry Pi.** 2013. Disponível em: <http://iqjar.com/jar/live-stream-from-your-raspberry-pi/>. Acesso em: 15 abr. 2014.

KAWAMOTO, Vinícius Ramos. **Desenvolvimento de aplicações para Android utilizando o framework Adobe Flex**. Medianeira, 2011. Trabalho de Conclusão de Curso (Graduação em análise e desenvolvimento de sistemas) - Universidade Tecnológica Federal do Paraná, Medianeira, 2011. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/533/1/MD_COADS_2011_2_08.pdf>. Acesso em: 08 jun. 2013.

LECHETA, Ricardo R.. **Google Android:** Aprenda a criar aplicações para dispositivos móveis com o Android SDK. 3. ed. São Paulo: Novatec, 2013.

MONK, Simon. **Adafruit's Raspberry Pi**: Lesson 4. GPIO Setup. 2013. 15p. Disponível em: <http://learn.adafruit.com/downloads/pdf/adafruits-raspberry-pi-lesson-4-gpio-setup.pdf>. Acesso em: 27 set. 2013.

MURATORI, José Roberto; DAL BÓ, Paulo Henrique. **Capítulo I - Automação residencial:** histórico, definições e conceitos. 2011. Disponível em: <http://www.osetoreletrico.com.br/web/documentos/fasciculos/Ed62_fasc_automacao_capI.pdf>. Acesso em: 01 set. 2013.

NASCIMENTO, Edmar José do. **Introdução às Redes de Computadores.** [2011]. Disponível em: <http://www.univasf.edu.br/~edmar.nascimento/redes/redes_20112_aula02.pdf>. Acesso em: 08 out. 2013.

NETTO, Daniel. **O uso do Raspberry Pi pelos profissionais de eletrônica**. Saber Eletrônica: Industrial, São Paulo, ano 48, n. 468, p.12-17, mar./abr. 2013. Bimestral. Disponível em: <http://www.revistapcecia.com.br/download/SE468_webS2.pdf>. Acesso em: 14 set. 2013.

OHA. **Android Overview**. [200-?]. Disponível em: <http://www.openhandsetalliance.com/android_overview.html>. Acesso em: 01 set. 2013.

OLIVEIRA, Rômulo Silva de; CARISSIMI, Alexandre da Silva; TOSCANI, Simão Sirineo. **Sistemas Operacionais**. 4ª. ed. Porto Alegre: Bookman, 2010. 11 v. (Série Livros Didáticos Informática UFRGS).

PEREIRA, Lúcio Camilo Oliveira; SILVA, Michel Lourenço da. **Android para Desenvolvedores**. Rio de Janeiro: Brasport, 2009.

PINTO, Pedro. **Transforme o seu Raspberry Pi num servidor FTP em 2 minutos.** 2013. Disponível em: <http://pplware.sapo.pt/linux/transforme-o-seu-raspberry-pi-num-servidor-ftp-em-2-minutos/>. Acesso em: 15 abr. 2014.

PROJETO ARDUINO**. Raspberry shield de automação**. [2013?]. Disponível em: <http://www.projetoarduino.com.br/raspberry-shield-de-automacao-p78>. Acesso em: 18 ago. 2013.

PRUDENTE, Francesco. **Automação Predial e Residencial:** Uma Introdução. Rio de Janeiro: Ltc, 2011. 211 p.

RASPBERRY Pi Exp. Board User Manual. [2013?]. Disponível em: <http://dl.dropboxusercontent.com/u/28708678/Raspberry%20pi%20Exp.pdf>. Acesso em: 18 ago. 2013.

RASPBERRY PI FOUNDATION. **About us**. [2011?]. Disponível em: <http://www.raspberrypi.org/about>. Acesso em: 15 set. 2013.

RASPBERRY WEB SERVER. **Using MySQL on a Raspberry Pi.** [2013?]. Disponível em: <http://raspberrywebserver.com/sql-databases/using-mysql-on-a-raspberry-pi.html>. Acesso em: 15 abr. 2014.

SILVEIRA, João Alexandre da. **Raspberry Pi:** O barramento GPIO. [2013?]. Disponível em: <http://www.ordemnatural.com.br/on_articles/raspberryPiGPIO.html>. Acesso em: 31 maio 2014.

TANENBAUM, Andrew S.. **Sistemas Operacionais Modernos**. 3 ed. São Paulo: Pearson Prentice Hall, 2009. 638 p.

WILLRICH, Roberto. **Linguagens de Programação.** [200-?]. Disponível em: <http://algol.dcc.ufla.br/~monserrat/icc/Introducao_linguagens.pdf>. Acesso em: 23 nov. 2013.

GLOSSÁRIO

Alt-get – APT é um gerenciador de pacotes para o sistema operacional Linux. Usa-se "apt-get" para baixar pacotes.

CD – Comando utilizado para mudar de diretório no terminal do Linux.

Driver – É um controlador que transmite e interpreta dados entre o sistema operacional e algum tipo de *hardware.*

Hardware – Parte física de um computador ou elemento eletrônico.

Install – Instalar. Quando usado em conjunto com o comando “apt-get install”, baixa e instala o pacote desejado no sistema operacional Linux.

Nano – Editor de texto do Linux. Pode ser comparado com o bloco de notas do Windows.

Relé – Interruptor eletromagnético, serve para ligar e desligar equipamentos.

Resistor – Dispositivo elétrico capaz de transformar energia elétrica em térmica ou limitar a corrente elétrica em um circuito.

Software – Sequência de instruções escritas para serem interpretadas por um computador ou dispositivo com o objetivo de executar uma tarefa.

Streaming - Transmissão de áudio e/ou vídeo através de uma rede de computadores, neste método o dispositivo repassa as informações ao mesmo tempo que recebe, sem a necessidade de *downloads*.

Sudo - Permissão de super usuário. Permite a usuários comuns obter privilégios de outro usuário, em geral do super usuário, afim de executar tarefas específicas dentro do sistema de maneira segura e controlável pelo administrador.

Terminal – Em informática é um equipamento disponibilizado ao usuário que serve de interface a um sistema de informação. No Linux, terminal é um interpretador de comandos utilizado para controlar a máquina, aplicativos e o próprio sistema operacional.

Thread – Forma de dividir um mesmo processo em duas ou mais tarefas executadas de maneira concorrente. Permite a execução de tarefas diferentes ao mesmo tempo e no mesmo aplicativo.