CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO

Yuri da Silva Chaves

SISTEMA DE ATENDIMENTO RESIDENCIAL IMPLEMENTANDO UMA SOLUÇÃO VOIP BASEADA EM ASTERISK

Timóteo

2021

Yuri da Silva Chaves

SISTEMA DE ATENDIMENTO RESIDENCIAL IMPLEMENTANDO UMA SOLUÇÃO VOIP BASEADA EM ASTERISK

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Lucas Pantuza Amorim

Timóteo

2021

Yuri da Silva Chaves

SISTEMA DE ATENDIMENTO RESIDENCIAL IMPLEMENTANDO UMA SOLUÇÃO VOIP BASEADA EM ASTERISK

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais, campus Timóteo, como requisito parcial para obtenção do título de Engenheiro de Computação.

Trabalho aprovado. Timóteo, 11 de setembro de 2021:

_____

Prof. Dr. LUCAS PANTUZA AMORIM
Orientador

_____

Prof. Dr. ELDER DE OLIVEIRA RODRIGUES
Professor Convidado

_____

Prof. Ms. ADILSON MENDES RICARDO
Professor Convidado

Timóteo
2021

Emitido em 11/09/2021

FOLHA DE ROSTO (PLATAFORMA BRASIL) Nº 1/2021 - DCCTM (11.63.05)

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 12/09/2021 16:03)
ADILSON MENDES RICARDO
PROFESSOR ENS BASICO TECN TECNOLOGICO
CECOMTM (11.51.22)
Matrícula: 2849338

(Assinado digitalmente em 13/09/2021 17:01)
ELDER DE OLIVEIRA RODRIGUES
PROFESSOR ENS BASICO TECN TECNOLOGICO
DCCTM (11.63.05)
Matrícula: 1694225

(Assinado digitalmente em 11/09/2021 16:53)
LUCAS PANTUZA AMORIM
PROFESSOR ENS BASICO TECN TECNOLOGICO
DCCTM (11.63.05)
Matrícula: 2897411

(Assinado digitalmente em 14/09/2021 16:06)
Yuri da Silva Chaves
DISCENTE
Matrícula: 201617060380

Para verificar a autenticidade deste documento entre em https://sig.cefetmg.br/documentos/ informando seu número: **1**, ano: **2021**, tipo: **FOLHA DE ROSTO (PLATAFORMA BRASIL)**, data de emissão: **11/09/2021** e o código de verificação: **1664fd47a3**

Agradecimentos

Agradeço à Deus pela vida e por sua graça que me trouxe até aqui. Agradeço a minha família que foi quem sempre esteve ao meu lado independente da situação.

Agradeço aos meus amigos do CEFET-MG pelas alegrias multiplicadas e tristezas divididas durante os períodos.

Agradeço aos meus colegas de trabalho, de *startup's* e outros projetos que me ajudaram a ser quem sou, que de alguma forma me ensinaram e me tornaram uma pessoa melhor.

Agradeço aos professores que me ajudaram a dar os primeiros passos nesse imenso mundo que é a computação.

Pessoas precisam de pessoas. Se estou aqui hoje é porque tive todos vocês me apoiando.

Muito obrigado.

“Não dispomos de pouco tempo, mas desperdiçamos muito”.
Sêneca

Resumo

Com o avanço da automação residencial e o surgimento da Internet das coisas, dispositivos embarcados podem ser integrados a um sistema de voz sobre IP (VOIP) para se comunicar com diversos dispositivos por meio de uma rede telefônica. Essa integração torna possível a comunicação entre um dispositivo embarcado e um telefone celular sem a necessidade de um *chip*. Este trabalho apresenta uma solução de automação residencial para interfones onde é realizada a comunicação entre um Raspberry-Pi e um aplicativo Android instalado no celular do proprietário através de um servidor VOIP Asterisk. O Raspberry-Pi atua como um interfone contendo um circuito onde um botão dispara uma requisição para o servidor Asterisk que por sua vez encaminha a chamada ao endereço IP registrado pelo aplicativo desenvolvido. Inicialmente foi levantado um estudo sobre a utilização de uma biblioteca da Amazon Alexa para realizar a ligação. Entretanto, a solução se mostrou inviável. Com a necessidade de realizar a comunicação entre o interfone e o proprietário, foi estudado e criado um servidor VOIP Asterisk. Para realizar e receber uma ligação foram criados dois clientes VOIPs: um cliente no Raspberry-Pi para realizar a ligação; e outro cliente sendo um aplicativo Android nativo que se autentica no servidor como o proprietário e recebe as ligações. Além disso, foi configurada uma rede VPN para que a comunicação aconteça de modo privado.

Palavras-chave: VOIP, automação residencial, VPN, interfone, Raspberry-Pi, Android.

Abstract

With the advancement of home automation and the emergence of the Internet of Things, embedded devices can be integrated with a voice over IP (VOIP) system to communicate with multiple devices over a telephone network. This integration makes communication between an embedded device and a mobile phone possible without the need for a *chip*. This work presents a home automation solution for intercoms where communication is performed between a Raspberry-Pi and an Android application installed on the owner's cell phone through an Asterisk VOIP server. The Raspberry-Pi acts as an intercom containing a circuit where a button triggers a request to the Asterisk server, which in turn forwards the call to the IP address registered by the developed application. Initially, a study was carried out on the use of an Amazon Alexa library to carry out the connection. However, the solution proved to be unfeasible. With the need to carry out communication between the intercom and the owner, an Asterisk VOIP server was studied and created. To make and receive a call, two VOIP clients were created: a Raspberry-Pi client to make the call; and another client being a native Android app that authenticates to the server as the owner and takes calls. In addition, a VPN network has been set up so that communication takes place privately.

Keywords: VOIP, home automation, VPN (Virtual Private Network), Intercom, Raspberry-Pi, Android.

Lista de ilustrações

Lista de tabelas

Lista de códigos

Lista de símbolos

GSM	*Global System for Mobile Communications*
IA	*Inteligência Artificial*
ARM	*Advanced RISC Machine*
API	*Application Program Interface*
AVS	*Alexa Voice Service*
IOT	*Internet Of Things*
PLN	*Processamento de Linguagem Natural*
ASR	*Automatic Speech Recognition*
JSON	*Java Script Object Notation*
HTTP	*Hypertext Transfer Protocol*
SDK	*Software Development Kit*
UDP	*User Datagram Protocol*
RTP	*Real-time Transport Protocol*
SIP	*Session Initiation Protoco*
PBX	*Private Branch Exchange*
PSTN	*Public Switched Telephone Network*
TLS	*Transport Layer Secutiry*
GPIO	*General Purpose Input/Output*

Sumário

1 Introdução

“Conheça-te a ti mesmo.”
Sócrates de Atenas

A automação residencial, também conhecida como domótica, corresponde a utilização das inovações tecnológicas que visam automatizar rotinas que trazem maior conforto e segurança dos moradores de determinada habitação. A palavra domótica tem origem na palavra latina “*Domus*” que significa “Casa”, unida à palavra "Robótica", que é a automatização e controle de qualquer processo. A área está em crescente evolução nas últimas décadas, auxiliada pelo avanço da tecnologia e aproximação da mesma com atividades ligadas ao cotidiano. Dessa maneira, a domótica permite ao usuário controlar dispositivos eletrônicos de sua residência através de interfaces de controle (EUZÉBIO, 2011).

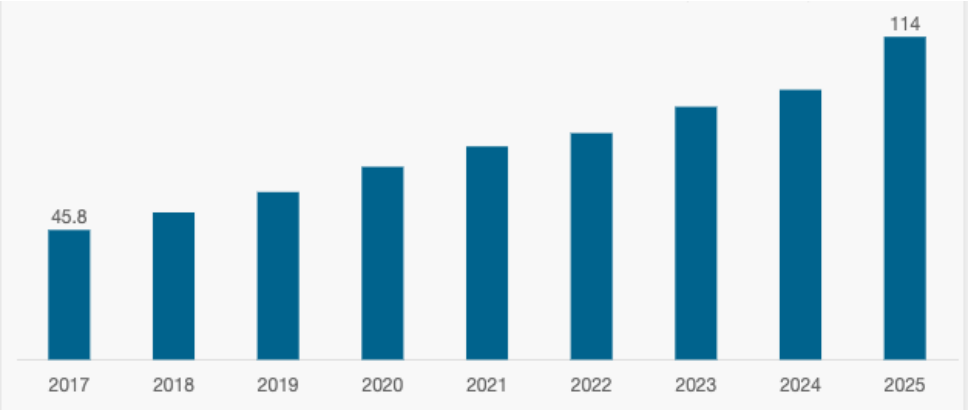
De todos os sistemas domóticos (de automação residencial), o de segurança patrimonial é um dos mais procurados pelos usuários atualmente. Em uma pesquisa realizada com 932 pessoas pela State of the Smart Home (HOME, 2015), cerca de 90% dos entrevistados concordaram que a segurança residencial está no topo da lista das características mais importantes da domótica.

Um estudo realizado em 2009 pelo Instituto Brasileiro de Geografia e Estatística (IBGE, 2010), demonstrou que 47,2% da população brasileira com 10 anos ou mais de idade sente-se insegura nas cidades em que moram e 21,4% da população no domicílio em que reside. Com esses dados observa-se que a segurança residencial tem importante visibilidade no mercado. De acordo com a Associação Brasileira de Automação Residencial (AURESIDE, 2016), cerca de 300 mil residências brasileiras possuem automação. Porém, esse número é considerado muito baixo em relação à quantidade de residências existentes. Um dos principais motivos é o alto custo e o desconhecimento das pessoas sobre o assunto. Logo, a expansão de residências automatizadas passa a ser um desafio para os desenvolvedores de novas tecnologias.

Em um estudo realizado em 2019 pela Fortune Business Insights (BUSINES, 2019b), o mercado global de automação residencial alcançou um valor de US$ 45,8 bilhões em 2017 e possui uma projeção de atingir US$ 114 bilhões até 2025 (Figura 1). A crescente demanda por sensores inteligentes, interruptores, relés, entre outros equipamentos inteligentes, o aumento da taxa de adoção de tecnologias avançadas como IA e NFC e o desenvolvimento de cidades inteligentes em todo mundo estão favorecendo o avanço no mercado de automação residencial (BUSINES, 2019a).

Uma aplicação típica da domótica é em segurança. O número de crimes de roubo e furto praticados em residências aumentou cerca de 20% nos últimos anos (IPEA, 2015). Para

Figura 1 – Tamanho do mercado global de automação residencial de 2016-2025 (US$ Bilhões).



Fonte: (BUSINES, 2019a).

verificar se a casa está vazia, os bandidos tocam o interfone, campainha ou telefonam para a residência para sondar o local. O interfone comum não permite ao residente saber quem chamou ou a hora que aconteceu, impedindo-o de tomar alguma decisão. A Tabela 1 mostra o monitoramento e segurança como uma das principais áreas que se destacam na busca pela automação residencial.

Tabela 1 – Utilização das principais tecnologias na automatização residencial.

TECNOLOGIA	2003	2004	2005	2015
Cabeamento estruturado	42%	61%	49%	80%
Monitoramento de segurança	18%	28%	29%	81%
Controle de iluminação	1%	2%	6%	75%
Automação integrada	0%	2%	6%	70%
Gerenciamento de energia	1%	5%	11%	62%

Fonte: (WILLIAN SANTOS RENATO, 2019).

Embora existam soluções comerciais como da empresa Alfa para comunicação entre a interface de um interfone residencial com um aparelho celular (SEGURANCA, 2019) e ainda, trabalhos que trazem diferentes resoluções para o tema (GUSMÃO, 2012) e (BAMPI, 2018), as tecnologias utilizadas para solucionar o problema se mostram ineficazes e obsoletas se comparado com as tecnologias atuais disponíveis graças ao avanço tecnológico. Além disso, são soluções proprietárias que não agregam valor a comunidade *open source*.

1.1 Justificativa

Transformar casas em confortáveis refúgios capazes de oferecer segurança e economia de custos é uma das vantagens da automação residencial (AURESIDE, 2016). Por questões de segurança, conforto, baixo custo e praticidade, será possível controlar quem tentou

acessar o condomínio ou casa, além disso a utilização de *softwares* livres em toda a solução, que reduz no custo de aquisição e manutenção do protótipo.

1.2 Objetivos

Esse trabalho tem como objetivo o desenvolvimento de um sistema para automação residencial para interfones que se comunique com um aplicativo Android através de um servidor VOIP. Ele poderá se comunicar com o morador quando o mesmo estiver dentro ou fora de casa através de um dispositivo de *hardware* e *software* que funciona como um interfone que se comunica com um aplicativo celular através de um servidor Asterisk.

1.2.1 Objetivos Específicos

1. Desenvolvimento de um servidor de comunicação entre o dispositivo de interfone e um aplicativo Android;
2. Desenvolvimento de um aplicativo para receber ligações VOIP e fornecer uma lista com histórico de visitantes;
3. Integração de um dispositivo de *hardware* à um serviço VOIP.

2 Fundamentação Teórica

“Viver significa lutar.”
Sêneca

Os dispositivos inteligentes possuem características da tecnologia chamada de Internet das Coisas (IOT), desenvolvida com o intuito de interligar objetos dentro de uma área determinada (AGRAWAL, 2013). A compatibilidade entre sistemas operacionais e dispositivos embarcados ampliou as possibilidades de prototipagem para o desenvolvimento na área de IOT. Neste capítulo, serão apresentados separadamente os equipamentos de *hardware*, sistemas e conceitos com o intuito de fornecer embasamento ao trabalho.

2.1 Domótica

A domótica é compreendida como uma implementação de tecnologias capazes de controlar e gerenciar dispositivos presentes em um ambiente e aplicação de técnicas de IA na predição e resolução de falhas. Um sistema domótico é dividido em vários subsistemas, cada qual atua especificamente em um campo de controle. Atualmente estes sistemas são informatizados e computadorizados e envolvem dispositivos e eletrodomésticos como, por exemplo os ilustrados na Figura 2 (SGARBI, 2006).

É uma área de estudos que visa melhorar a qualidade de vida, reduzir o trabalho doméstico, aumentar o bem estar e a segurança de seus habitantes bem como a utilização racional e planejada dos diversos recursos.

Figura 2 – Aplicações de uma domótica.

Fonte: (DECORTIPS, 2021).

2.2 Raspberry-Pi

Raspbarry-Pi é um computador de baixo custo, pequeno e portátil. Ele pode ser utilizado para plugar em um monitor ou televisão, teclado, mouse, *pendrive* etc. O Raspberry-Pi possui um sistema integrado que permite aos usuários programar sistemas com integração com componentes de *hardwares* analógicos e digitais. Além disso, os programadores podem desenvolver *scripts* ou programas em diversas linguagens de programação (ZHAO JAYANAND JEGATHEESAN, 2015). A Tabela 2 descreve a evolução e comparativo dos modelos disponíveis no mercado.

Tabela 2 – Modelo de placa Raspberry-Pi 3 B+.

	Raspberry Pi A+	**Raspberry Pi model B**	**Raspberry Pi 2**	**Raspberry Pi Zero**	**Raspberry Pi Zero W**	**Raspberry Pi 3**	**Raspberry Pi 3 model B+**	**Raspberry Pi 4 Model B**
Lançamento	10/11/2014	15/02/2012	01/02/2015	30/11/2015	28/02/2017	29/02/2016	2018	2019
Preço US$	US$20.00		US$35.00	US$5.00	US$10.00	US$35.00	US$35.00	US$35.00 a US$55.00
Tipo Chip	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2836	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2837	Broadcom BCM2837B0	Broadcom BCM2711
Tipo Core	ARM1176JZF-S	ARM1176JZF-S	Cortex-A7	ARM1176JZF-S	ARM1176JZF-S	Cortex-A53 64-bit	Cortex-A53 (ARMv8) 64-bit	Cortex-A72 (ARM v8) 64-bit
Nº Core	1	1	4	1	1	4	4	4
Clock CPU	700 MHz	700 MHz	900 MHz	1 GHz	1 GHz	1.2 GHz	1.4 Ghz	1.5GHz
GPU	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore VI
RAM	256 MB	512 MB (256 - model A)	1 GB	512 MB	512 MB	1 GB	1GB LPDDR2 (900 MHz)	2GB, 4GB ou 8GB
Wireless	X	X	X	X	802.11n	802.11n	802.11n 2.4GHz/5.0GHz	802.11ac 2.4 GHz/5.0 GHz
Bluetooth	X	X	X	X	4.1 (BLE)	4.1 (BLE)	4.2 (BLE)	5.0 (BLE)
Consumo	180 mA	500 mA	350 mA	100 mA	150 mA	350 mA	500 mA	600 mA

Fonte: (HUTCHINSON, 2019).

Uma característica importante do Raspberry-Pi é a utilização da arquitetura ARM de seu processador, que fornece suporte a diversos sistemas operacionais como: Raspbian, Ubuntu, Windows 10 entre outros. O modelo utilizado neste trabalho é o Raspberry-Pi 3 B+ com o sistema operacional Ubuntu. Recomenda-se utilizar o modelo B+ para projetos embarcados, pois oferece maior flexibilidade, exigem baixo consumo de energia e possuem mais portas USB (FOUNDATION, 2021).

2.3 VOIP

Voz sobre IP (VOIP) é uma tecnologia que permite que chamadas telefônicas atravessem redes regulares baseadas em IP. Em seu núcleo, as redes VOIP são construídas em dois tipos de protocolos: sinalização e mídia. No contexto de uma rede VOIP, um protocolo de sinalização é uma linguagem que lida com a configuração, desmontagem e controle de chamadas, como por exemplo, durante a ligação de um telefone `A` para o telefone `B`, eles devem esta-

belecer um idioma comum para discutir informações, entre si ou entre cada um dos sistemas telefônicos locais, como:

- Qual é a origem e o destino da chamada;
- Quais *codecs* (maneiras de codificar dados de voz e vídeo) cada lado suporta;
- Saber se outro lado desligou e completou a chamada.

Os protocolos de sinalização tratam de todas essas questões. O protocolo de sinalização mais comum é o SIP (*Session Initiation Protocol*). Esse protocolo enquadra-se na camada de aplicação do modelo de referência OSI e pode ser executado no TCP (*Transmission Control Protocol*) ou no UDP (U*ser Datagram Protocol*). Em sua estrutura, o protocolo SIP implementa métodos de requisição e resposta na comunicação, podendo transmitir qualquer informação independendo da camada de transporte. Por ser um protocolo bastante difundido nos ambientes de multimídia, ele é amplamente utilizado em soluções VOIP (CRITELLI, 2020e).

Os protocolos de mídia tratam do transporte de áudio e vídeo codificados em tempo real que são transmitidos entre os terminas. A medida em que cada pacote de mídia chega ao outro lado da conversa ele é decodificado e reproduzido. O SIP não suporta qualquer tipo de mídia em seus pacotes e para a transmissão de mídia é utilizado o RTP (*Real-time Transport Protocol*). O RTP provê funções de transporte de rede fim-a-fim ajustadas para aplicações transmitindo dados em tempo-real, como áudio, vídeo ou dados simulados, sobre serviços de rede (GROSS, 2009). O RTP é executado em cima do UDP. Para transmissão em tempo real, recomenda-se o protocolo UDP pois ele não lida com retransmissão de pacotes perdidos evitando problemas de bloqueio de chamada para retransmissão dos pacotes (CRITELLI, 2020e).

2.4 Amazon Alexa

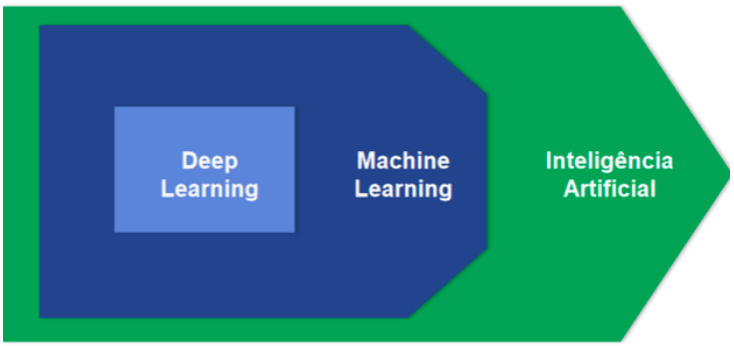
Alexa é um serviço de voz baseado em nuvem da Amazon disponível em centenas de milhões de dispositivos da Amazon e de outros fabricantes de dispositivos. Com Alexa, é possível criar experiências de voz naturais que oferecem aos clientes uma maneira mais intuitiva de interagir com a tecnologia que usam todos os dias. Uma das formas se obter diferentes ações é através dos comandos de voz que disparam habilidades dentro da Alexa. É possível solicitar, por exemplo, que a Alexa chame um Uber, peça comidas pelo iFood, exiba resultados, dentre outros serviços (AMAZON, 2020b).

2.4.1 Inteligência Artificial

Artificial, diz respeito ao que não é natural, feito pelo homem para imitar a natureza (MICHAELIS, 2020). Já a inteligência não possui uma definição exata. Pode se dizer que está relacionada com a capacidade de resolver situações novas com rapidez e êxito, adaptando-se a ela por meio do conhecimento adquirido (MICHAELIS, 2020). A partir do desse conceito, tem-se como Inteligência Artificial (IA) a criação de máquinas capazes de aprender ou reconhecer

padrões, fazendo uso de algoritmos que proporcionem a tomada de decisão, especulações, e interações com base nos dados recebidos. Entretanto, IA pode ser subdividida em camadas (Figura 3) que a compõem, e a partir disso temos os conceitos de *Machine Learning* e *Deep Leaning* (DAMACENO, 2018).

Figura 3 – Camadas de inteligência artificial.

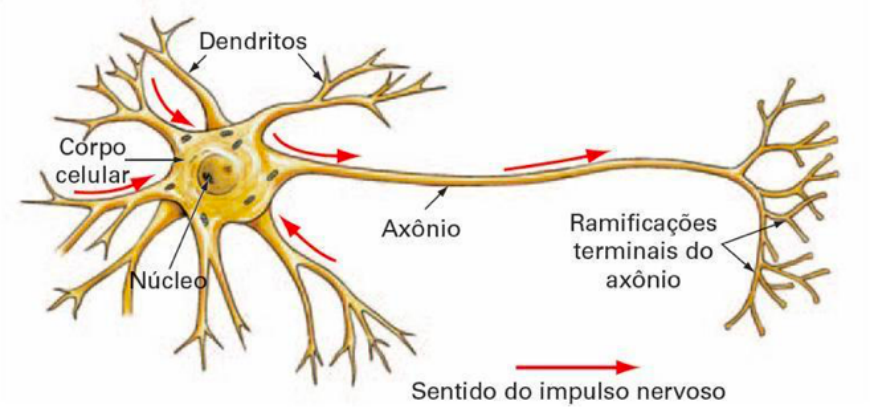


Fonte: Campos (2017).

Fundamentalmente, *Machine Learning* é a aplicação de algoritmos com a intenção de extrair informação de um conjunto de dados e expressá-los por meio de algum modelo matemático. Este modelo é então utilizado para fazer a dedução a partir do conjunto de dados. Diversos algoritmos permitem fazer isso, mas um tipo em especial se destaca, o de redes neurais artificiais (ACADEMY, 2018a).

A unidade fundamental de uma rede neural artificial é um nó (neurônio matemático), que por sua vez é baseado no neurônio biológico. Os neurônios originais, embora existam várias formas deles, transmitem um sinal elétrico de uma extremidade a outra, dos dendritos ao longo dos axônios até os terminais (Figura 4). Esses sinais são então passados de um neurônio para outro (RASHID, 2016).

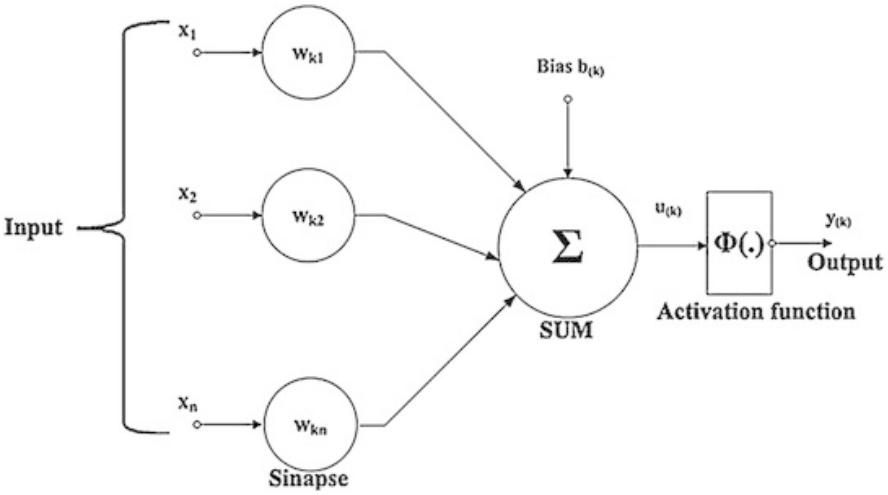
Figura 4 – Representação simplificada do neurônio biológico.



Fonte: (ACADEMY, 2018a).

Já um neurônio matemático de uma rede neural artificial é um componente que calcula a soma ponderada de várias entradas, aplica uma função e passa o resultado adiante (ACADEMY, 2018a). A Figura 5 ilustra o funcionamento de um neurônio matemático.

Figura 5 – Representação simplificada do neurônio matemático.



Fonte: (ACADEMY, 2018a).

2.4.2 Processamento de Linguagem Natural (PLN)

Um desafio de longa data dentro da inteligência artificial tem sido construir máquinas inteligentes, e uma parte importante do comportamento inteligente é entender a linguagem natural. O Processamento de Linguagem Natural (PLN) é um ramo da Inteligência Artificial projetado para ajudar as máquinas a entender o processo natural de comunicação dos seres humanos (ACADEMY, 2018b). As aplicações típicas em PNL incluem reconhecimento de fala, compreensão de linguagem falada, sistemas de diálogo, análise lexical, análise, tradução automática, gráfico de conhecimento, recuperação de informações, resposta a perguntas, análise de sentimento, computação social, geração de linguagem natural e resumo de linguagem natural (DENG, 2018).

A principal força motriz por trás da PLN é o *Deep Leaning*. As abordagens utilizadas pelo *Deep Leaning* exploram as poderosas redes neurais que contêm várias camadas para resolver tarefas gerais de aprendizado de máquina, reduzindo o trabalho especializado de uma pessoa para separação, escolha e normalização adequada dos dados a serem processados (DENG, 2018). Os sistemas de reconhecimento automático de fala fazem uso das técnicas da PNL de uma forma bastante restrita baseadas em gramáticas. As técnicas de PNL são úteis em ASR ao modelar a linguagem natural ou o domínio de interação através da produção de um conjunto preciso de regras para a gramática onde as estruturas para a linguagem são definidas (TRILLA, 2009).

2.4.3 Habilidades

As habilidades da Alexa são chamadas de *skills* e possuem um modelo de interação e lógica de aplicativo. Quando um cliente fala, a Alexa processa a fala no contexto de seu modelo de interação para determinar a solicitação do cliente. Alexa envia então a solicitação para a lógica de aplicação de sua habilidade. A lógica de cada habilidade é disponibilizada como um serviço de nuvem dentro de servidores da Amazon que são chamados de AWS ou outros servidores (AMAZON, 2020a).

Figura 6 – Como Alexa funciona.

Fonte: (AMAZON, 2020b)

O serviço de voz da Alexa utiliza ASR (*Automatic Speach Recognition*) e trata do processamento de liguagem natural (PLN), descrito na Seção 2.4.2, para compreensão da fala e invocação da *skill* que o usuário deseja. Cada *skill* possui seu modelo de interação por voz que definem as palavras e frases que determinam e controlam suas *skills* (AMAZON, 2020b). Alexa possui suporte para dois tipos de modelos de interação:

- **Modelo pré construído:** conjunto de palavras predefinidas onde o usuário apenas escolhe qual *skill* será invocada pela solicitação. Por exemplo: “Alexa, acenda a luz” ou “Alexa, desligue a televisão”;
- **Modelo personalizado:** oferece maior flexibilidade, porém sua configuração gera maior complexidade. Nela o usuário configura todas as maneiras de solicitar uma *skill*. Por exemplo: “Alexa, planeje uma viagem de São Paulo para o Rio de Janeiro”, “Alexa, quero ir de São Paulo ao Rio de Janeiro” e “Alexa, planeje uma viagem ao Rio de Janeiro”.

Para o desenvolvimento de uma *skill* é necessário disponibilizar um *end point* (url) acessível na internet para hospedar o serviço que será requerido pela *skill* e um ambiente de desenvolvimento apropriado para a linguagem de programação que deseja criar a lógica da *skill*. O serviço pode ser criado em qualquer linguagem apropriada para o desenvolvimento *web*. No processo de criação da *skill*, a Amazon disponibiliza ferramentas e bibliotecas de desenvolvimento de *software* através da *Alexa Skill Kit* (ASK), que fornecem acesso programático aos recursos da Alexa permitindo com que se gaste mais tempo na implementação de recursos e menos tempo escrevendo código trivial (AMAZON, 2020b). Essa ferramenta é disponibilizada em Node.js (FOUNDATION, 2021), Java (ORACLE, 2021), e Python (ORGANIZATION, 2021).

2.4.4 Serviço de Voz da Amazon (AVS)

Assim como o serviço da Alexa capacita os dispositivos da Amazon Echo, a AVS disponibiliza esse mesmo serviço a fabricantes de dispositivos integrar um conjunto cada vez maior de recursos e funções do Alexa em um produto conectado. Fabricantes de engenharia originais (OEM), fabricantes de design original (ODM) e integradores de sistemas (SI) usam AVS para transformar Alexa em alto-falantes, fones de ouvido, computadores, televisores, veículos e produtos domésticos inteligentes (AMAZON, 2020c).

Alguns dispositivos com AVS possuem um conjunto de microfone habilitado para receber comandos de voz, alto-falante e um conjunto básico de recursos e funções da Alexa. Para personalizar esses comandos e criar novas funções é fornecido um conjunto de bibliotecas em C++ que integra rapidamente quase todos os recursos e funções da Alexa de forma modular e abstrata.

3 Trabalhos Relacionados

A automação residencial brasileira apresenta números muito inferiores se comparados a outros países em função do valor elevado, baixa usabilidade e principalmente quando se quer integrar algum dispositivo ou serviço terceiro no sistema residencial. Com isso, o mercado brasileiro apresenta potencial de expansão no quesito automação residencial (SILVA, 2018).

3.1 Automação Residencial com IA

O trabalho de Caio-Vanderlei (SILVA, 2018) objetivou verificar, de forma conceitual através de uma automação residencial inteligente e conectada, utilizando os recursos de IOT e Inteligência Artificial, a percepção de qualidade e satisfação dos serviços de automação residencial apresentados no mercado brasileiro. Um dos pontos relevantes apontados é que a domótica como IOT não é o suficiente para promover ações consideradas inteligentes. No entanto, a união de dispositivos eletrônicos e Inteligência Artificial apresenta uma solução com maior capacidade de interação entre os dispositivos controlados e o morador em relação aos trabalhos de domótica mais simples apresentados no mercado. Como o experimento é somente conceitual, foi elaborada uma maquete de uma casa representando os recursos e funções interligados com a assistente pessoal Alexa da Amazon.

Para a coleta de informações nos cômodos, foram inseridos sensores e atuadores que fornecem dados para os controladores. As luzes são controladas pelo aplicativo Blink (BLYNK, 2021) e pelos comandos de voz da assistente pessoal Alexa (AMAZON, 2020a).

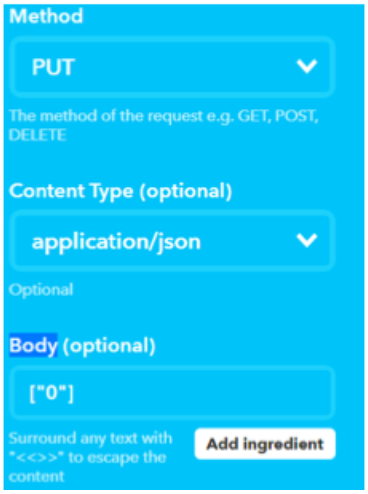
O aplicativo escolhido foi pensado para que os botões físicos da casa também fossem virtuais, de tal forma que possa acionar os atuadores de qualquer lugar do mundo através de requisição HTTP. Para o acionamento via assistente pessoal Alexa, foram configuradas algumas frases que disparam requisições para a API IFTTT (IFTT, 2021) que se comunica com o aplicativo Blink. A Figura 7 apresenta a configuração de requisição de uma palavra ou frase.

3.2 Protótipo de Automação Residencial Utilizando uma Assistente de Voz

Com a finalidade de trazer maior praticidade para o usuário efetuar atividades rotineiras em uma residência, Leandro (NETO, 2018) propôs a criação de um protótipo de automação residencial utilizando o microcontrolador Arduino Ameba integrando-o com a assistente pessoal Alexa através dos serviços *web* (AWS) da Amazon.

Neste trabalho foram pontuadas como variáveis de controle três itens: controle de uma lâmpada, controle da temperatura ambiente da casa e abertura de uma porta eletrônica. Como pode se notar na Figura 8, após a fala do usuário é enviada uma requisição JSON para os

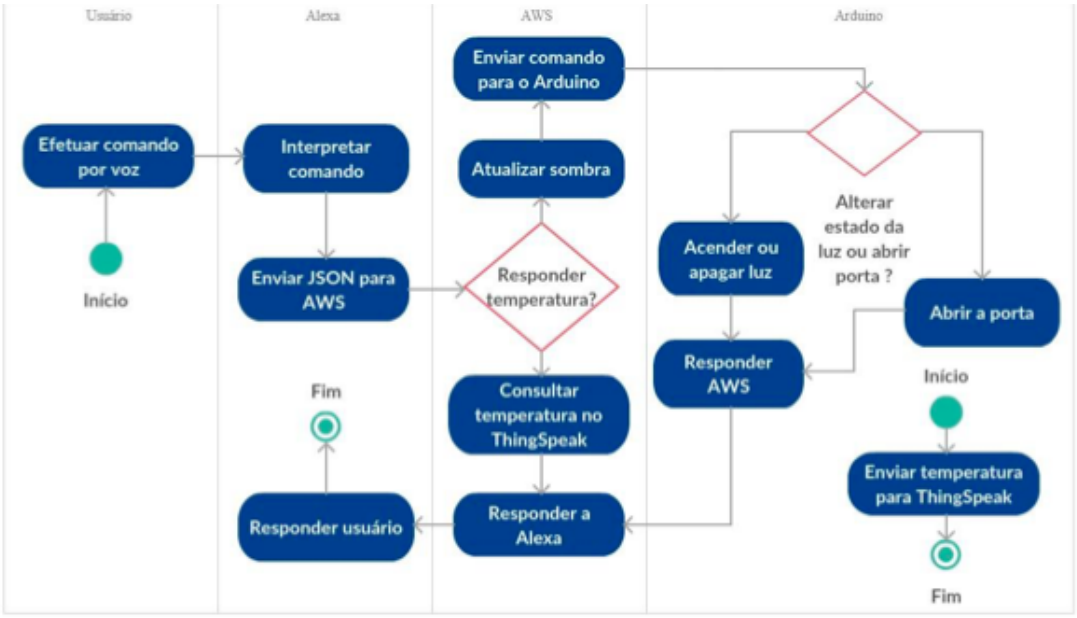
Figura 7 – Configuração do protocolo e sinalização.



Fonte: (SILVA, 2018).

servidores da Amazon onde está hospedada a lógica da aplicação, que é responsável por buscar os dados de temperatura que são enviados constantemente pelo Arduino ao AWS e armazenados, ou enviar um comando para o Arduino que por sua vez irá decidir se o comando é para alterar o estado da luz ou para abrir o portão.

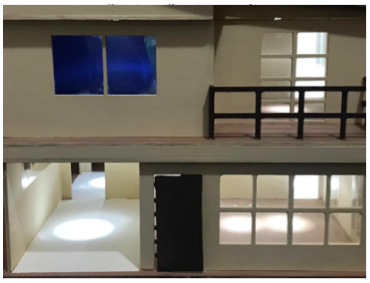
Figura 8 – Diagrama de atividades.



Fonte: (NETO, 2018).

Como resultado foi apresentado uma maquete (Figura 9) com luzes sendo representadas por componentes *LED*. O controle da abertura da porta é feito por um eletroímã de 12V que, quando energizado, mantém a porta fechada e, quando a alimentação é cortada, a porta é aberta.

Figura 9 – Imagem frontal da maquete.



Fonte: (NETO, 2018)

3.3 Campainha de Vídeo *Dorbell*

A empresa Ring (RING, 2021) é uma multinacional que pertence a Amazon e é conhecida por comercializar aparelhos que auxiliam na segurança residencial. Um de seus principais produtos é uma campainha para identificação de visitantes com recursos de vídeo, chamados Dorbell (INC, 2021). A Amazon, sendo detentora de todos os direitos da empresa, anunciou que será adicionada a assistente pessoal Alexa aos produtos da empresa dando-lhe a capacidade de conversar com os visitantes da residência equipada. Os principais ganhos com essa nova tecnologia são:

- Alexa poderá perguntar aos seus visitantes o que eles desejam;
- Os visitantes poderão optar por deixar uma gravação de uma mensagem por vídeo;
- Caso seja recebimento de uma encomenda, a Alexa poderá dizer onde deixar a encomenda;
- Os moradores poderão programar uma mensagem pronta, como por exemplo "não podemos atender a porta agora, mas se você quiser deixar uma mensagem, pode fazê-lo agora".

Além disso, a Amazon traz para algumas de suas campainhas e câmeras um sensor de movimento que alerta audivelmente ao visitante que ele está sendo gravado quando se aproxima do equipamento. Todos esses recursos podem ser ativados ou desativados pelo aplicativo Ring (INC, 2021).

4 Materiais e métodos

“O correr da vida embrulha tudo. A vida é assim: esquenta e esfria, aperta e daí afrouxa, sossega e depois desinquieta. O que ela quer da gente é coragem.”
Guimarães Rosa

O presente trabalho aborda automação de uma portaria. Trata-se de uma pesquisa descritiva, aplicada e experimental, envolvendo diferentes dispositivos, linguagens, *softwares* e tecnologias entre si.

As especificações dos dispositivos e sistemas utilizadas são:

***Hardware*:**

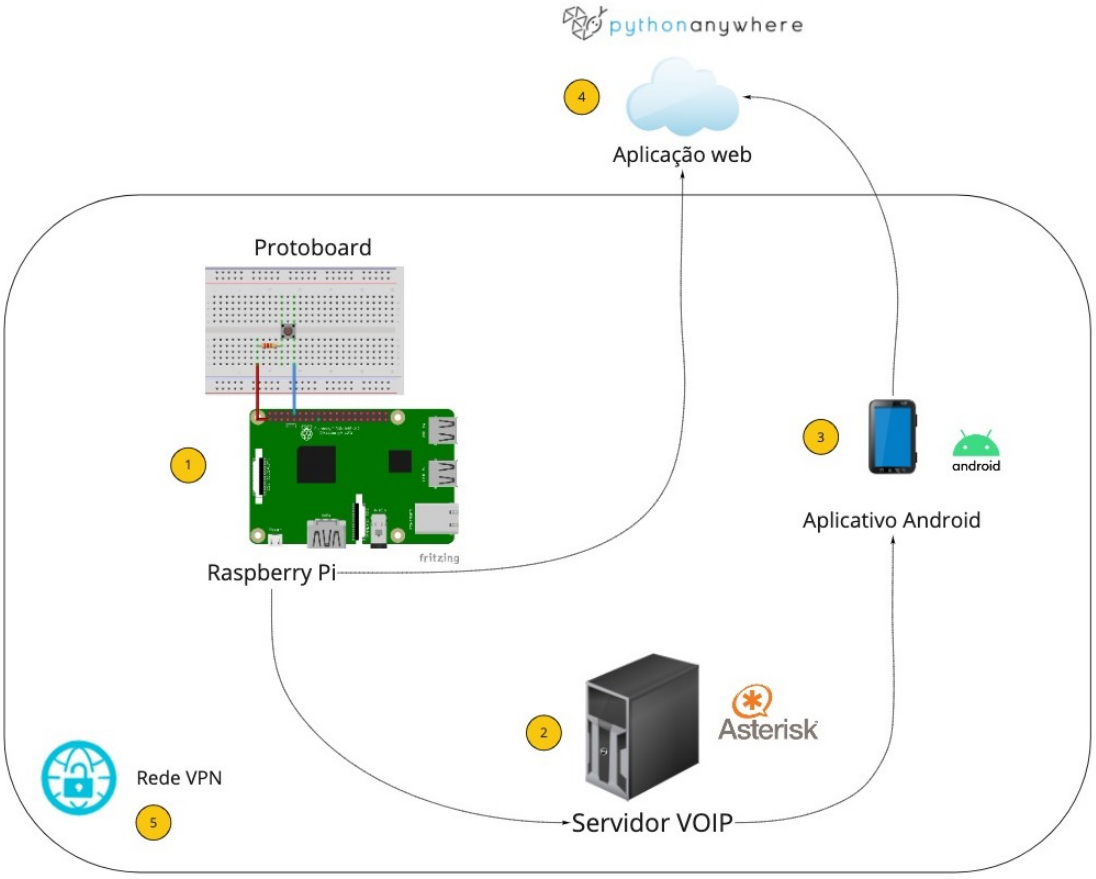
- Raspberry-Pi 4 modelo B com memória RAM de 2GB e 4 portas USB (ORG, 2021).
- Microfone USB 2.0 *plug and play*;
- Samsung Galaxy J5 Prime com Android 8;
- Macbook Air 8GB de RAM e processador Intel core i5, utilizado somente para acessar o Raspberry-Pi remotamente;
- Fios de ligação (*jumpers*);
- *Protoboard*;
- Resistor de 220 Ohms 5%;
- Botão interruptor tátil;
- Mini caixa de som Kp-600 - Knup.

***Software*:**

- Sistema operacional Ubuntu 16.04 LTS (UBUNTU, 2021);
- Asterisk (BRYANT, 2021);
- Android Studio versão 4.2.1 (DEVELOPERS, 2021a);
- PythonAnywhere para serviço *web* (LLP, 2021).

Os procedimentos utilizados para desenvolver esse projeto foram seguidos visando entender o funcionamento de cada tecnologia e sua aplicação na prática desejada. A Figura 10 ilustra os dispositivos e *softwares* utilizados e a interação entre eles. As seções a seguir descrevem o processo de pesquisa e desenvolvimento de cada item da Figura 10.

Figura 10 – Diagrama metodológico do projeto.



Fonte: Elabora pelo autor.

4.1 Comunicação VOIP

Dentre as etapas citadas, uma das principais é a comunicação VOIP. O servidor VOIP será responsável por fazer a integração entre o interfone e o telefone celular do proprietário. A tecnologia VOIP foi muito difundida e bem recebida pelas corporações desde seu lançamento, pois é muito atrativo migrar o serviço de telefonia para trafegar sobre a rede de computadores que já engloba grande parte das empresas e uma parcela significativa das residências do mundo (GROSS, 2009).

No contexto de uma rede VOIP, os protocolos são responsáveis pela configuração, desmontagem e controle das chamadas. O protocolo é a linguagem comum que os dispositivos utilizam para se comunicar. Eles tratam do transporte do áudio ou vídeo codificado que é enviado entre os terminais.

O protocolo mais comum em uso como protocolo de mídia nas redes modernas é o protocolo de transporte em tempo real (RTP). O RTP é executado em cima do UDP (*User Datagram Protocol*). O UDP não é orientado à conexão e não lida com a retransmissão de

pacotes perdidos.

A plataforma para construção da aplicação de comunicação em tempo real neste trabalho foi o Asterisk (MALCOM, 2021) utilizando o protocolo SIP. O SIP fornece uma implementação mais simples em comparação com os outros protocolos, além disso ele possui alguns princípios subjacentes:

Simplicidade: inclui apenas seis métodos;

Independência da camada de transporte: ele pode usar UDP, TCP, ATM, etc.

Mobilidade pessoal dos usuários: os usuários podem se trocar de rede sem quaisquer restrições devido à atribuição de um identificador exclusivo para cada usuário;

Escalabilidade da rede : a estrutura de rede baseada no protocolo SIP facilita a expansão e o aumento do número de componentes;

Extensibilidade : o protocolo caracteriza-se pela possibilidade de complementá-lo com novas funcionalidades quando surgirem novos serviços;

Integração na pilha de protocolos da Internet existente: o SIP é parte de uma arquitetura multimídia global desenvolvida pela IETF. A arquitetura também inclui os protocolos RSVP, RTP, RTSP e SDP.

O Asterisk é um sistema telefônico (PBX) IP que roda como um servidor em diversos sistemas operacionais como Linux, Windows, Mac ou BSD. O Asterisk possui todas as funcionalidades de outros sistemas telefônicos do mercado, com a vantagem de ser *open source* (código aberto) e, portanto pode ser utilizado sem qualquer custo ou necessidade de licença (GROSS, 2009).

Como precisamos de interoperabilidade para comunicação entre o Raspberry-Pi e o aplicativo Android, o Asterisk é uma excelente opção, pois faz a conversão entre diversos *codecs* e protocolos do mercado e ainda elimina a obrigação de grande parte dos módulos externos que providenciam recursos avançados fundamentais na maioria das centrais telefônica, como: música em espera, correio de voz e URA (Unidade de Resposta Audível) (GROSS, 2009). O mercado já conta com várias soluções baseadas no aplicativo para as mais diversas atividades de negócios, agregando maior valor e confiabilidade a um projeto de VoIP baseado em *software* livre.

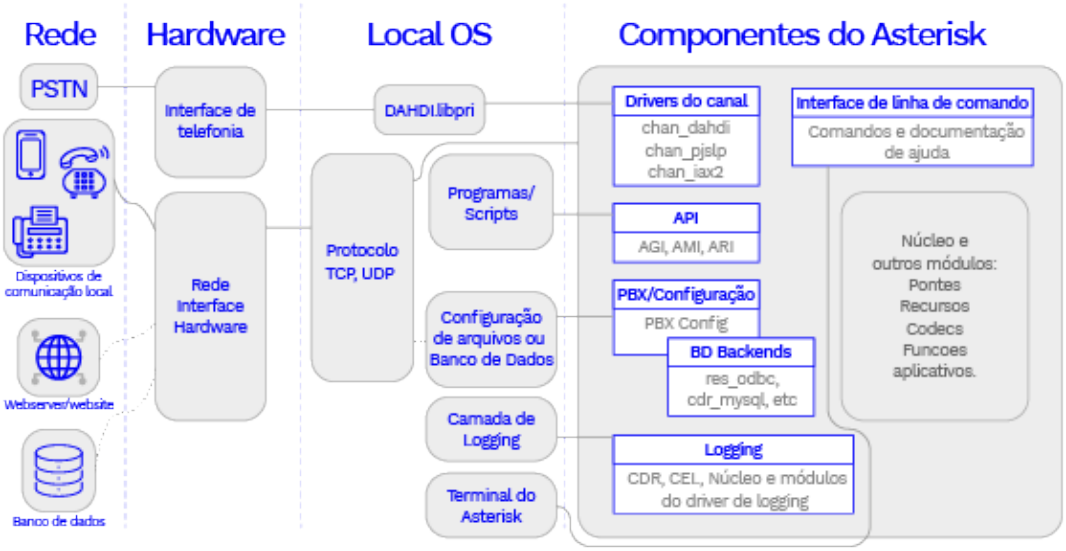
Uma das características mais importantes do Asterisk é sua arquitetura extremamente modular. Nem todo sistema VoIP precisa aproveitar todos os recursos que o Asterisk pode oferecer. Essa arquitetura permite habilitar somente os módulos necessários em seu projeto, promovendo a independência entre as partes do seu sistema VoIP (CRITELLI, 2020c).

O Asterisk é um sistema adaptável devido sua estrutura interna embasada por quatro APIs (*Application Programming Interface*) próprias em torno do “sistema núcleo central”.

Este trata as conexões internas do PABX desconsiderando protocolos, *codecs*, interfaces telefônicas e aplicativos que dão a possibilidade de usar qualquer *hardware* e qualquer tecnologia)(CRITELLI, 2020d).

A Figura 11 é um diagrama simplificado que ilustra os relacionamentos de alguns componentes principais entre si e com entidades fora do Asterisk (MALCOM, 2021). É útil entender como um componente pode se relacionar com coisas fora do Asterisk, já ele não está normalmente operando sem alguma conectividade ou interação com outros dispositivos de rede ou arquivos no sistema local (NEWTON, 2014a).

Figura 11 – Um sistema Asterisk.



Fonte: (NEWTON, 2014a).

4.1.1 Núcleo do Asterisk

O núcleo do PBX é o componente essencial que fornece a infraestrutura. Entre muitas funções, ele lê os arquivos de configuração, incluindo *dialplan* (Seção 4.1.4) e carrega todos os outros módulos (Seção 4.1.2), componentes distintos que fornecem mais funcionalidades. O núcleo carrega e constrói o *dialplan*, que é a lógica de qualquer sistema Asterisk contendo uma lista de instruções que deve ser seguida para se saber como lidar com entrada e saída de chamadas no sistema (NEWTON, 2014a).

4.1.2 Módulos do Asterisk

Além da funcionalidade fornecida pelo núcleo do Asterisk, os módulos fornecem todas as outras funcionalidades. Os códigos fonte de muitos módulos são fornecidos pelo Asterisk, embora outros módulos possam ser disponibilizados por membros da comunidade ou mesmo empresas que fazem módulos comerciais. Os módulos nativos do o Asterisk podem ser opcionalmente compilados quando o Asterisk for inicializado.

A maioria dos módulos são configuráveis independentemente e têm seus próprios arquivos de configuração. Do ponto de vista logístico, esses módulos são normalmente arquivos com uma extensão de arquivo `.so` e são localizados no diretório de módulos do Asterisk (que normalmente é /usr/lib/asterisk/modules). Quando o Asterisk é inicializado, ele carrega esses arquivos e adiciona sua funcionalidade ao sistema (NEWTON, 2014a).

Existem muitos tipos diferentes de módulos, cada um fornecendo suas próprias funcionalidades e capacidades para o Asterisk. Alguns exemplos de módulos são:

***Drivers* de canal:** os *drivers* de canal se comunicam com dispositivos fora do Asterisk e traduzem aquela sinalização ou protocolo específico para o núcleo;

Aplicativos *dialplan*: os aplicativos fornecem funcionalidade de chamada para o sistema. Um aplicativo pode atender uma chamada, reproduzir um efeito sonoro, desligar uma chamada ou fornecer um comportamento mais complexo, como enfileiramento, correio de voz ou conjuntos de recursos de conferência;

Funções *dialplan*: as funções são usadas para recuperar, definir ou manipular várias configurações em uma chamada. Uma função pode ser usada para definir o ID do chamador em uma chamada de saída, por exemplo;

***Codecs*:** um *codec* (que é um acrônimo para COder / DECoder) é um módulo para codificação ou decodificação de áudio ou vídeo. Normalmente, os *codecs* são usados para codificar mídia para que ocupe menos largura de banda. Eles são essenciais para traduzir o áudio entre os *codecs* de áudio e os tipos de carga usados por diferentes dispositivos.

Esse trabalho utilizou principalmente no módulo PJSIP *Channel Driver*. Um *driver* de canal permite que o Asterisk interaja com terminais externos. O *driver* de canal PJSIP habilita o Asterisk a lidar com *endpoints* SIP, como o telefone SIP (VoIP) que será configurado.

4.1.3 Chamadas e canais

O objetivo principal do Asterisk é ser uma plataforma para a construção de sistemas e aplicativos de comunicação em tempo real. Na maioria dos casos, isso implica dizer que o conceito de “ligações” aparece com frequência. As chamadas, na terminologia de telefonia, geralmente se referem a um telefone se comunicando com (chamando) outro telefone por um meio, como uma linha PSTN (telefonia tradicional). No entanto, no caso do Asterisk, uma

chamada normalmente faz referência a um ou mais canais existentes no (NEWTON, 2014a). São alguns exemplos de “chamadas”:

- Um telefone ligando para outro telefone pelo Asterisk;
- Um telefone chamando vários telefones ao mesmo tempo através do Asterisk;
- Um telefone ligando para um módulo ou o inverso. (Por exemplo, correio de voz);
- Um canal local interagindo com um aplicativo ou outro canal.

Vale ressaltar que telefones podem se referir a qualquer canal ou grupo de canais como uma chamada. Não importa se são telefones ou qualquer outro dispositivo, como um sensor de sistema de alarme, controle de porta de garagem ou um sistema embarcado (NEWTON, 2014a).

Alguns tipos de canais comumente usados:

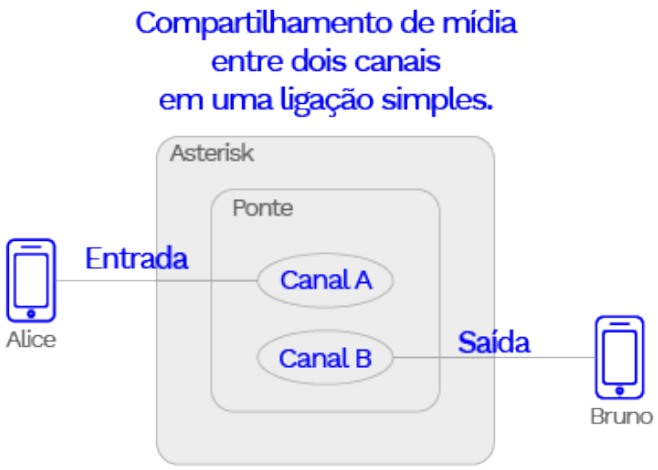
- Um *driver* de canal SIP como `chan_sip` ou `chan_pjsip`;
- Canais DAHDI fornecidos por `chan_dahdi`;
- Canais locais fornecidos por `chan_local`.

Quase nada acontece no Asterisk sem um canal envolvido. Um canal é uma entidade dentro do Asterisk que atua para promover a comunicação entre o Asterisk e outro dispositivo, ou seja, um telefone, um PBX, outro sistema Asterisk ou mesmo o próprio Asterisk (no caso de um canal local). Uma chamada pode ser um ou mais canais, criando um caminho de comunicação ou atividade através do sistema. Um dos benefícios do Asterisk é a capacidade de interagir com tantas tecnologias diferentes quantos *drivers* de canal configurados. No entanto, a maioria dos administradores usará apenas alguns tipos de cada vez (NEWTON, 2014b).

Depois que um canal é estabelecido, os eventos que ocorrem dependem da tecnologia do canal. Ou seja, se a comunicação de áudio, vídeo ou outros dados começa no canal dependerá da sinalização que ocorre no SIP, ISDN, H.323 ou outros protocolos implementados por meio do *driver* do canal (NEWTON, 2014b).

Quando o Asterisk estabelece um canal, ele usará uma combinação de configuração de *driver* de canal e instrução *dialplan* para determinar como o canal se comporta. Além disso, o Asterisk pode se comunicar com programas externos de forma síncrona ou assíncrona para receber chamadas de API para inspeção, direção ou manipulação de canal. Na Figura 12 o canal de entrada é criado a partir do telefone de Alice ligando para o Asterisk. O Asterisk então liga para o ramal discado por Alice criando um canal de saída para falar com Bruno. Uma vez

Figura 12 – Criação de canais em uma ligação simples.



Fonte: (NEWTON, 2014b).

que a chamada é estabelecida, os dois canais são colocados em uma ponte que compartilha a mídia entre eles (NEWTON, 2014b).

Para este trabalho foram utilizados os canais SIP que fazem interface com dispositivos VOIP com capacidade SIP, como telefones, bancos de canais, outros PBXs ou provedores de serviços de telefonia por IP.

4.1.4 Dialplan

O *dialplan*, ou plano de discagem, é essencialmente uma linguagem de *script* específica para o Asterisk e uma das principais formas instrução sobre como se comportar, permitindo que seja encaminhada e manipulada chamadas de maneira programática. O Dialplan é composto por arquivos de texto (por exemplo `extensions.conf`) escrito em linguagem de *script* nos formatos AEL ou LUA. Alternativamente, o *dialplan* pode ser lido de um banco de dados junto com a configuração de outro módulo. Com o *dialplan*, haverá uso intenso de aplicativos e funções para afetar canais, configuração e recursos (JORDAN, 2015).

O *dialplan* é responsável pelo roteamento das chamadas, por isso é frequentemente referido como o coração de um sistema Asterisk. Embora existam outras interfaces de programação para interagir com o Asterisk, o *dialplan* é o mais básico, e entendê-lo é fundamental para entender como o Asterisk trata as chamadas. De acordo com MEGGELEN RUSSELL BRYANT, existem quatro componentes fundamentais para o plano de discagem do Asterisk:

- **Extensões**: uma extensão é simplesmente um agrupamento de etapas usadas para lidar

com uma chamada específica. Ao contrário de muitos sistemas telefônicos tradicionais, os ramais não precisam ser numéricos e não estão vinculados a um único dispositivo.

- **Contextos**: um contexto é uma seção lógica no *dialplan*. Os contextos contêm uma ou mais extensões.
- **Prioridades**: uma prioridade é uma etapa de uma extensão. As prioridades tratam do pedido e também podem ter rótulos anexados para que uma chamada possa alternar entre as prioridades conforme necessário.
- **Aplicativos**: os aplicativos são muito semelhantes às funções das linguagens de programação tradicionais. Eles dizem ao Asterisk o que fazer com uma chamada. Por exemplo, você pode ter uma extensão interna de `*86` que quando é discado, pode fazer com que o Asterisk reproduza a mensagem do dia usando o aplicativo Playback (Aplicativo dentro do Asterisk que retorna uma mensagem gravada).

No arquivo `extensions.conf` são cadastradas as rotas que determinam quais chamadas serão feitas a partir de uma requisição enviada ao servidor Asterisk.

4.2 Instalando Asterisk no Linux

Depois de entender os fundamentos do VoIP e da plataforma Asterisk, foi instalado e configurado o Asterisk no Raspberry-Pi. Até então, o Asterisk não distribui oficialmente pacotes para distribuições Linux, então é necessário compilar o Asterisk a partir do seu código-fonte.

O primeiro passo foi baixar e descompactar uma versão atual do Asterisk. A versão utilizada foi a 16.6.1. O processo de construção e execução foi facilitado através do *script* `install_prereq` para instalar automaticamente as dependências necessárias com base na sua distribuição. Foi executado o *script* no modo `teste`, que imprime o comando necessário instalar as dependências, como mostra a Figura 13. Em seguida, foi executado o *script* no modo `instalar` para realmente instalar as dependências necessárias (CRITELLI, 2020b).

Com os os pré-requisitos instalados, é possível executar os *scripts* de configuração em preparação para a construção do Asterisk. O comando abaixo irá executar scripts de configuração para o Asterisk:

```
1  $ ./configure --with-jansson-bundled
```

Por padrão, o Asterisk usa o `menuselect` utilitário para apresentar uma lista gráfica de opções de configuração como mostra a Figura 14. Neste trabalho, foram deixadas todas as opções padrão, com uma exceção da desmarcação do módulo `chan_sip` no menu *drivers* de canal. O módulo `chan_sip` é um *driver* de canal SIP mais antigo e obsoleto e não é necessário em um ambiente Asterisk moderno (CRITELLI, 2020b).

Com o *script* de configuração executado, foi construído o Asterisk usando o comando `make`. Assim que a compilação foi concluída, foi executado `make install` para realizar a instalação. Como recomendado pela documentação (MALCOM, 2021), utilizou-se os comandos

Figura 13 – Comando instalação Asterisk.

```
[root@asterisk-1 asterisk-16.6.1]# cd scripts

[root@asterisk-1 scripts]# ./install_prereq test
###############################################
## test: test mode.
## Use the commands here to install your system.
###############################################
yum install --skip-broken --assumeyes gcc-c++ libedit-devel jansson-
devel libuuid-devel sqlite-devel libxml2-devel speex-devel speexdsp-
devel libogg-devel libvorbis-devel alsa-lib-devel portaudio-devel
libcurl-devel xmlstarlet bison flex postgresql-devel unixODBC-devel
neon-devel gmime-devel lua-devel uriparser-devel libxslt-devel
openssl-devel mysql-devel bluez-libs-devel radcli-devel freetds-devel
jack-audio-connection-kit-devel net-snmp-devel iksemel-devel
corosynclib-devel newt-devel popt-devel libical-devel spandsp-devel
libresample-devel uw-imap-devel binutils-devel libsrtp-devel gsm-
devel doxygen graphviz zlib-devel openldap-devel hoard codec2-devel
fftw-devel libsndfile-devel unbound-devel subversion bzip2 patch
python-devel

[root@asterisk-1 scripts]# ./install_prereq install
```

Fonte: Elaborado pelo autor.

make samples para gerar amostras de arquivos de configuração e `make config` para gerar sistemas de arquivos de unidade (CRITELLI, 2020b).

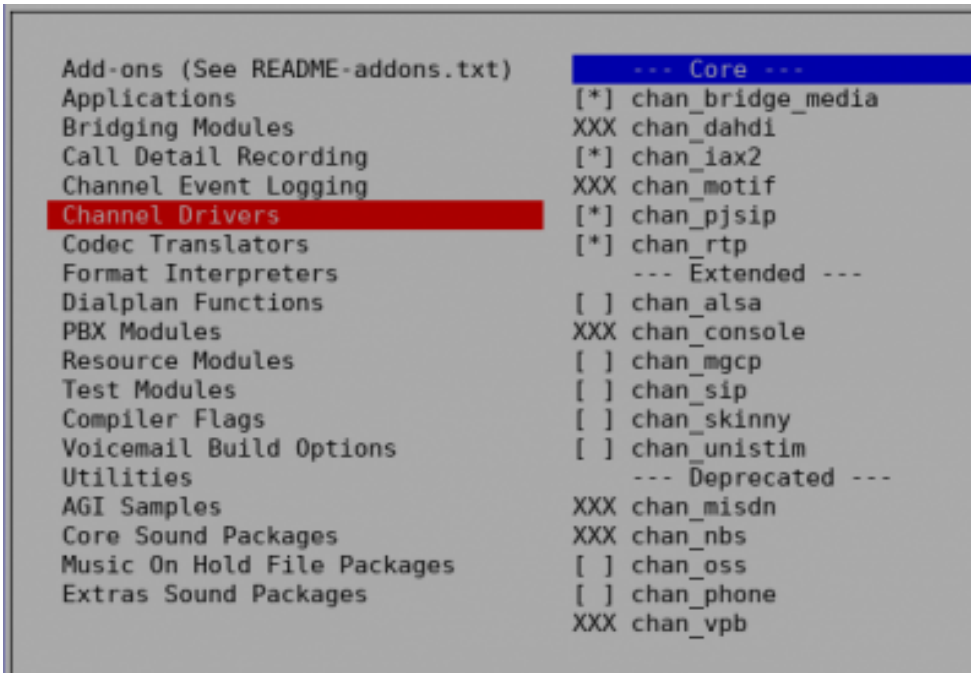
4.2.1 Configuração de arquivos

Antes de detalhar a configuração do Asterisk, é importante conhecer alguns arquivos e diretórios de configuração importantes. A Tabela 3 contém alguns diretórios que foram modificados no projeto.

4.2.2 Configurando PJSIP

Existem módulos para diversas funções, como lidar com correio de voz, conectar-se a bancos de dados externos e lidar com vários tipos de codificação de mídia. A primeira configuração feita foi implementar uma topologia de comunicação simples (Figura 15) utilizando o *driver* de canal PJSIP. Este *driver* de canal habilita o Asterisk a lidar com *endpoints* SIP como, por exemplo, os telefones que irão se conectar ao servidor. Por exemplo, o Asterisk pode fazer a ponte entre um *driver* de canal SIP e um *driver* de canal de rede telefônica pública comutada (PSTN) para permitir que seus telefones VoIP liguem para telefones das empresas de telefonia tradicionais.

Figura 14 – Menu de configuração do Asterisk.



Fonte: Elaborado pelo autor.

Figura 15 – Topologia de telefone simples.

Fonte: (CRITELLI, 2020a).

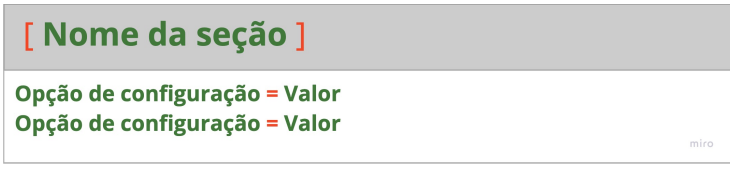
Inicialmente, o Asterisk precisa de uma configuração básica para PJSIP armazenada no arquivo `/etc/asterisk/pjsip.conf`. Este arquivo de texto é composto de seções, assim como a maioria dos arquivos de configuração usados no Asterisk. Cada seção define um objeto de configuração em `res_pjsip` ou um módulo associado. As seções são identificadas por nomes entre colchetes e possuem uma ou mais opções de configuração que podem ser atribuídas a um valor usando um sinal de igual seguido por um valor. Essas opções e valores são a configuração de um determinado componente de funcionalidade fornecido pelos respectivos módulos do objeto de configuração como mostra a Figura 16.

Tabela 3 – Arquivos e diretórios de configuração do Asterisk.

Arquivo	**Descrição**
`/etc/asterisk`	Este é o diretório base que contém os arquivos de configuração. Esse diretório foi preenchido com arquivos de configuração de amostra para muitos componentes do Asterisk quando executamos o comando `make samples` durante a instalação.
`/etc/asterisk/asterisk.conf`	Este arquivo contém a configuração básica do Asterisk. Notavelmente, ele contém os diretórios que o Asterisk usará para certas funções, como sons e *logs*.
`/etc/asterisk/modules.conf`	Este arquivo informa ao Asterisk quais módulos carregar. Como o Asterisk é modular é possível desligar os módulos que não está usando. Por padrão, todos os módulos do diretório `/usr/lib/asterisk/modules` são carregados (por meio da `autoload=yesdiretiva`).
`/etc/asterisk/pjsip.conf`	Este arquivo contém a configuração do *driver* do canal PJSIP.
`/etc/asterisk/extensions.conf`	Este arquivo é o *dialplan* (Seção 4.1.4). O Asterisk considera uma extensão uma coleção de instruções do *dialplan*. As extensões não estão vinculadas a telefones físicos. Eles são apenas um lugar no *dialplan* que contém o código que você deseja executar.

Fonte: (CRITELLI, 2020c)

Figura 16 – Sintaxe para objetos de configuração res_sip.



Fonte: Elaborado pelo autor.

A configuração básica, obtida diretamente da configuração de exemplo, é apenas o

suficiente para o PJSIP escutar na porta UDP padrão 5060 para SIP. Esta configuração informa ao *driver* do canal PJSIP para criar um transporte UDP vinculado a todos os endereços IP. O `res_pjsip` suporta opções de configuração para protocolos como TCP, UDP ou WebSockets e métodos de criptografia como TLS / SSL (CRITELLI, 2020a). Nesse trabalho foi utilizado o protocolo UDP, como mostra a Figura 17.

Figura 17 – Configuração básica do arquivo `pjsip.conf`.

```
[transport-udp]
type=transport
protocol=udp
bind=0.0.0.0
```

Fonte: (CRITELLI, 2020a).

Com a camada de transporte estabelecida, temos um PJSIP escutando em uma porta, mas esse fato não adianta muito sem ter a configuração adequada para aceitar registros de telefones. Além dessa configuração básica, o telefone requer a configuração de algumas seções. Cada telefone precisará de um dos seguintes:

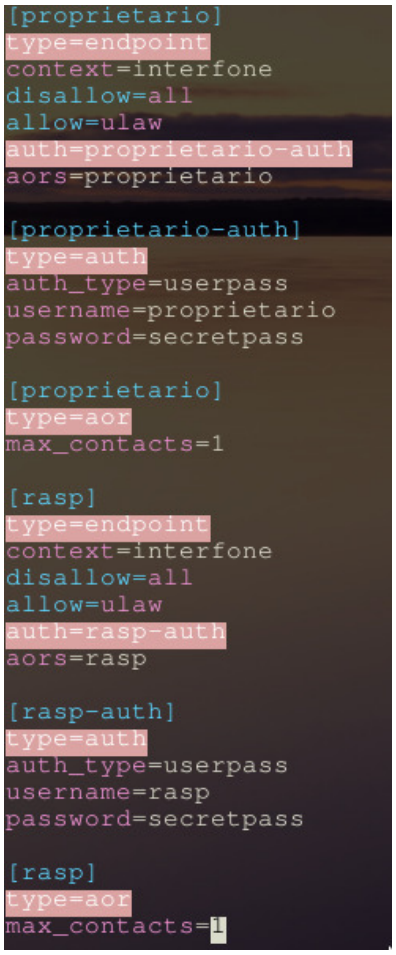
- ***Endpoint***: *endpoints* unem as outras partes da configuração SIP. É um terminal como uma entidade com a qual deseja entrar em contato, como o telefone de um usuário;
- ***Auth***: Armazenam dados de autenticação, como nome de usuário e senha para um *endpoint* SIP;
- ***Address of Record* (AOR)**: AOR diz ao Asterisk onde um *endpoint* pode ser contatado. Por exemplo, o endpoint “proprietario” pode ser acessado no endereço IP 10.100.100.150.

Com isso, foram configuradas as seções para os dois telefones no arquivo `/etc/asterisk/psjip.conf`. A Figura 18 ilustra o arquivo contendo as seções configuradas para cada usuário.

A primeira seção é uma seção de *endpoint* conforme denotado pelo `type`. Ele diz ao Asterisk que as chamadas de entrada devem ser colocadas no `context` “interfone”, e o `ulaw-codec G.711` é o único *codec* permitido para este terminal. O *endpoint* “proprietario” deve usar `proprietario-auth` para autenticação e o *endpoint* `rasp` utilizar o “rasp-auth” bem como cada um utilizando seus respectivos `AOR's`. As seções `auth` e `AOR` são bastante autoexplicativas. A opção `max_contacts` no `AOR` diz ao Asterisk que apenas um dispositivo (por exemplo um telefone) pode ser registrado por vez neste `AOR`.

Esta configuração não especifica um ramal numérico para o telefone do usuário. O Asterisk desacopla os terminais reais (telefones físicos, aplicativos que fazem chamada telefônica etc) da extensão. No Asterisk, um ramal é simplesmente um código identificado de forma única que é executado no plano de discagem (*dialplan*) do Asterisk quando uma chamada é

Figura 18 – Arquivo `pjsip.conf`.



Fonte: Elaborado pelo autor.

recebida (CRITELLI, 2020a).

Em seguida foi criado um *endpoint* real, como um número de telefone, para comunicar com o Asterisk. Com isso temos o servidor Asterisk configurado e o *endpoint* SIP pronto para receber as requisições de chamadas.

4.2.3 Configurando o *dialplan*

Após ter configurado o *driver* de canal PJSIP para conectar a um cliente de *softphone* (aplicativo de ligação telefônica pela Internet) simples com a instalação do Asterisk, foi configurado o *dialplan* (Seção 4.1.4), que conecta os canais no Asterisk. Isso porque os telefones ainda não podem ligar um para o outro sem essa conexão e ainda não foi dado a eles “extensões”(identificadores) numéricas.

O dialplan é configurado em `/etc/asterisk/extensions.conf`. A Figura 19 mostra as configurações necessárias para permitir que o interfone e o celular liguem um para o outro.

Figura 19 – Arquivo de configuração de rotas.

```
[interfone]
exten => 1001,1,Dial(PJSIP/proprietario)
exten => 1002,1,Dial(PJSIP/rasp)
```

Fonte: Elaborado pelo autor.

Segundo a Figura 19:

- **Interfone**: é o contexto. Este contexto possui duas extensões;
- **1001 e 1002**: são as extensões. Essa configuração separa os números dos próprios telefones. Embora esses números estejam configurados para ligar para o proprietário e para o Raspberry-Pi, eles poderiam ser facilmente reajustados para uma rotina mais complexa;
- **Número 1**: é a prioridade. Uma prioridade é apenas uma etapa no tratamento da extensão. A primeira prioridade é sempre `1`. Neste exemplo, cada ramal possui apenas uma única prioridade;
- **Dial**: é o aplicativo de discagem que é usado para ligar para um dispositivo remoto. A discagem ocorre via SIP ou outros protocolos de sinalização.

Quando a chamada entra no contexto “interfone”, o Asterisk tenta corresponder aquela chamada a um ramal. Quando o ramal `1001` é discado, o primeiro passo (prioridade) diz ao Asterisk para discar para o *endpoint* PJSIP do telefone do proprietário. Quando o ramal `1002` é discado, a mesma coisa acontece com o Raspberry-Pi.

Com o *dialplan* configurado e suas mudanças implementadas, o servidor está pronto para direcionar as chamadas. Os próximos passos foram configurar o *softphone* no Raspberry-Pi e no aplicativo do propretário.

4.3 Configurando *softphone* no Linux (Twinkle)

Para fazer a ligação a partir do Raspberry-Pi foi escolhido o *softpone* Twinkle (TWINKLE, 2017). Twinkle é um aplicativo de código aberto gratuito para telecomunicações de voz. Este programa foi criado especialmente para o sistema Linux e pode ser usado como telefone IP para telecomunicações VOIP. Os fluxos de mídia estão passando por RTP. Dentre suas principais características estão (FREEZVON, 2017):

- Encaminhamento de chamadas;
- Correio de voz;
- Chamada em espera;

- Mensagem instantânea;
- Rejeição de chamada;
- Chamada de conferência.

A instalação e inicialização do Twinkle foi realizada via terminal:

```
$ sudo apt-get install twinkle
$ twinkle
```

Após a inicialização, foram utilizadas as credenciais criadas no Asterisk para o usuário do Raspberry-Pi (Figura 18). Além do aplicativo com interface gráfica, o Twinkle fornece suporte via interface de linha de comando. Para realizar a ligação via linha de comando utilizou-se os comandos:

```
$ twinkle -c
$ call +551001
```

O comando `twinke -c` inicializa a aplicação do Twinkle e o comando `call` liga para o número desejado.

4.4 Criando aplicativo Android

Para a parte de comunicação com o celular do dono da casa, foi criado um aplicativo nativo utilizando a plataforma Android Studio versão 4.2.1 (DEVELOPERS, 2021a) e a linguagem Kotlin (DEVELOPERS, 2021c). O Android Studio é o ambiente de desenvolvimento integrado (IDE, na sigla em inglês) oficial para o desenvolvimento de apps Android. As dependências do projeto são especificadas por nome no arquivo `build.gradle`. Este arquivo se encarrega de encontrar as dependências e disponibilizá-las na versão. A dependência escolhida para usar a comunicação VOIP no Android foi a biblioteca Linphone (DEVELOPERS, 2021d).

4.4.1 Linphone

Linphone é uma biblioteca SIP de código aberto para chamadas de voz/vídeo e mensagens instantâneas, e está disponível para ambientes móveis e de *desktop* (iOS, Android, GNU / Linux, macOS, Windows). A Figura 20 ilustra o código para adicionar a biblioteca ao projeto no Android Studio.

O aplicativo faz login com o servidor Asterisk utilizando as credenciais fornecidas ao proprietário para que assim o servidor possa saber qual endereço IP deverá ser chamado quando for solicitado a ligação para o proprietário.

Figura 20 – Importanto SDK Linphone.

```
implementation 'org.linphone:linphone-sdk-android:5.0+'
// Adding this dependency allows the linphone-sdk to automatically handle audio focus
implementation 'androidx.media:media:1.3.1'
```

Fonte: Elaborado pelo autor.

4.4.2 Configuração do aplicativo

O processo para criar o aplicativo foi:

1. Importar biblioteca na pasta `build.gradle`;
2. Configurar uma tela para usuário fazer o *login*;
3. Listar visitas com a data e hora em que tocaram o interfone;
4. Cria tela para atender/recusar chamada VOIP;
5. Realizar testes de comunicação.

O aplicativo conta com 3 telas principais: (i) uma tela de login, para o usuário se autenticar junto ao servidor Asterisk informando seu usuário, senha e o endereço do servidor; (ii) uma tela contendo a listagem com data e hora de todas as vezes que o aplicativo recebeu uma ligação VOIP; (iii) e uma tela para controle de chamadas, para receber, recusar, finalizar e ou colocar/tirar do mudo uma chamada quando necessário.

Para realizar o *login* é necessário ainda escolher o protocolo de comunicação com o servidor, entre eles UDP, TCP ou TLS. O protocolo utilizado nesse trabalho foi o UDP, pois é o protocolo recomendado para comunicação em tempo real e permite uma comunicação rápida por não fornecer garantia na entrega dos pacotes (CRITELLI, 2020e).

Após ter efetuado o *login*, é aberta a tela inicial contendo uma lista de todas as visitas. Essa lista busca informações de um *webserver* via requisição `http` utilizando a biblioteca `Retrofit` (SQUARE, 2021). O `Retrofit` é um cliente REST de tipo seguro para Android, Java e Kotlin que facilita o consumo de serviços da *web*. Essa biblioteca torna o *download* de dados JSON ou XML de uma API da *web* bastante simples.

4.5 Configurando circuito GPIO no Raspberry-Pi

Para simular o toque do interfone foi criado um circuito conectado ao Raspberry-Pi que lê o estado de um botão. Para esse circuito foi utilizado os seguintes componentes eletrônicos:

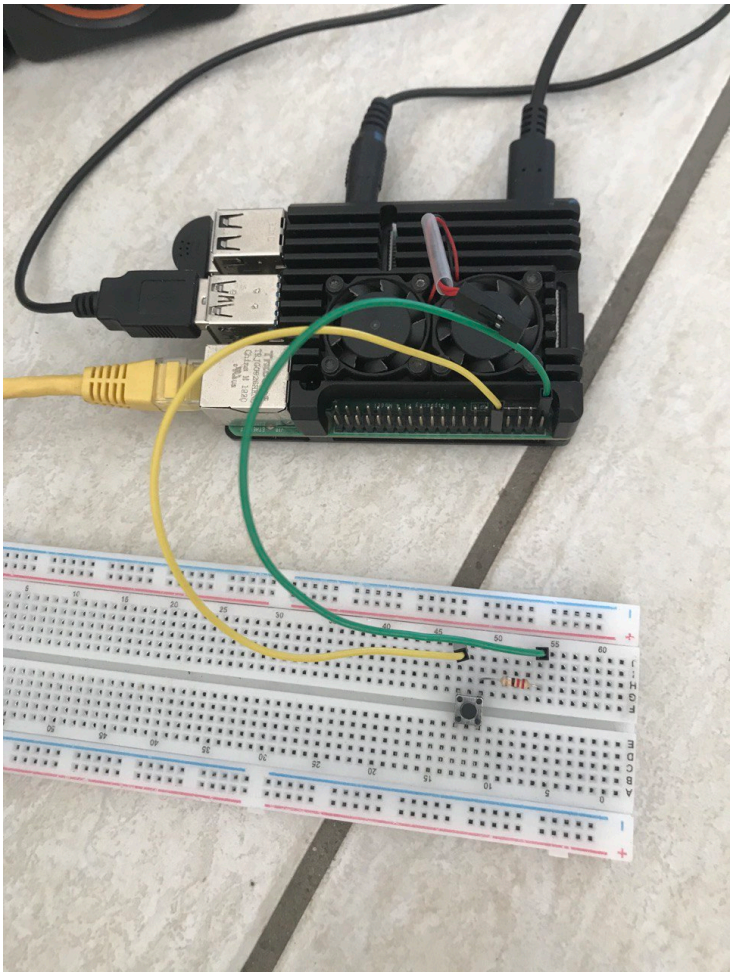
- *Protoboard*;
- Fios de ligação (*jumpers*);
- Resistor;

- Botão/ interruptor tátil.

Um dos fios de ligação recebe a alimentação de `3.3V` no `pino 1` do Raspberry-Pi. O outro fio é conectado a um pino de entrada do Raspberry-Pi (foi usado o pino `10`). Foi utilizado um resistor para limitar a corrente e proteger o pino de entrada, limitando a corrente que pode fluir. A Figura 21 ilustra o circuito montado.

O pino de entrada é baixo quando o botão não for pressionado (`0V`). Quando o botão é pressionado, ele conecta o pino a `3,3V` mudando o estado para alto.

Figura 21 – Circuito no Raspberry-Pi.



Fonte: (HQ, 2018).

Com o circuito criado, foi escrito um *script* na linguagem Python que lê o estado do botão e executa o código com base no estado. Foi necessário instalar um biblioteca do Python que permite acessar a porta GPIO diretamente. O comando a seguir foi executado no terminal do Raspberry-Pi:

```
1  $ sudo apt-get install python-rpi.gpio python3-rpi.gpio
```

Com a biblioteca instalada, foi importada e configurada a biblioteca GPIO para usar a numeração da placa. Foi inicializado o pino 10 como um pino de entrada e configurado o Raspberry-Pi a pressionar o pino para baixo usando os parâmetros `pull_up_down`. O código 4.1 contém a configuração do botão.

```
1  import RPi.GPIO as GPIO # Importa biblioteca Raspberry Pi GPIO
2
3  GPIO.setwarnings(False) # Ignora avisos
4  GPIO.setmode(GPIO.BOARD) # Usa n mero de PIN f sico
5  GPIO.setup(10, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) # Delega PIN 10 para ser a
       entrada do PIN e configura o valor inicial como desligado
6  GPIO.add_event_detect(10,GPIO.RISING,callback=button_callback) # Configura a
       fun   o a ser executada na borda de subida do PIN 10
7
8  def button_callback(channel):
9      print("Botao foi pressionado!")
10
11 message = input("Pressione qualquer tecla para sair! \n\n") # Executa o programa
       enquanto algu m  n  o aperta alguma tecla
12 GPIO.cleanup() # Limpa os recursos de entradas GPIO
```

Listing 4.1 – Código para ler a ação do botão

4.6 Rede VPN

Para parte da comunicação entre o Raspberry-Pi e o aplicativo Android através do servidor Asterisk foi configurada uma VPN (*Vitual Private Network*) utilizando um servidor Ubuntu server versão 18.04 para hospedar o serviço Open VPN (OPENVPN, 2021). Essa configuração foi feita para proteger o servidor Asterisk e suas portas de acesso remoto. Os servidores Asterisk são muito visados por *hackers* e para disponibilizar esse servidor na *web*, seria necessário tomar precauções em relação a segurança. Com a rede VPN, podemos atravessar redes não confiáveis com privacidade e segurança como se estivesse em uma rede privada. O tráfego inicia a partir do servidor VPN e continua seu trajeto para o destino (ELLINGWOOD, 2021).

O Open VPN (OPENVPN, 2021) é uma solução VPN de código aberto *Secure Socket Layer* (SSL) que possui ampla gama de configurações. Para instalar e utilizar o Open VPN foram necessários:

- Servidor Ubuntu 18.04 para hospedar seu serviço OpenVPN;
- Aplicação Open VPN;
- Emissor de certificados confiáveis (CA) (JETHA, 2018);
- Um aplicativo para se autenticar na rede.

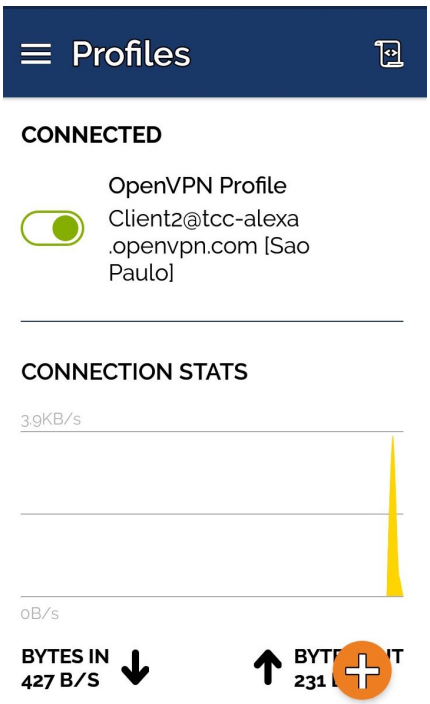
Foram criados dois arquivos de configuração para os clientes OpenVPN: um para o cliente Raspberry-Pi e um para o aplicativo Android. Os arquivos fornecem todas as configurações necessárias para se conectar ao servidor VPN criado, possuindo credenciais e acessos. A utilização desse arquivo foi feita usando o aplicativo OpenVpn que importa o arquivo criado e se conecta ao servidor VPN.

Para o Raspberry-Pi foi gerado o arquivo `client1.ovpn`. O comando utilizado para se conectar ao servidor VPN é:

```
1  $ sudo openvpn client1.ovpn
```

Por parte do celular Android foi gerado um segundo arquivo chamado `client2.ovpn` contendo as credenciais para se conectar ao servidor. Para a conexão foi utilizado o aplicativo OpenVPN para Android, disponível na loja de aplicativos Play Store (DEVELOPERS, 2018) como ilustra a Figura 22.

Figura 22 – Aplicativo Open VPN.



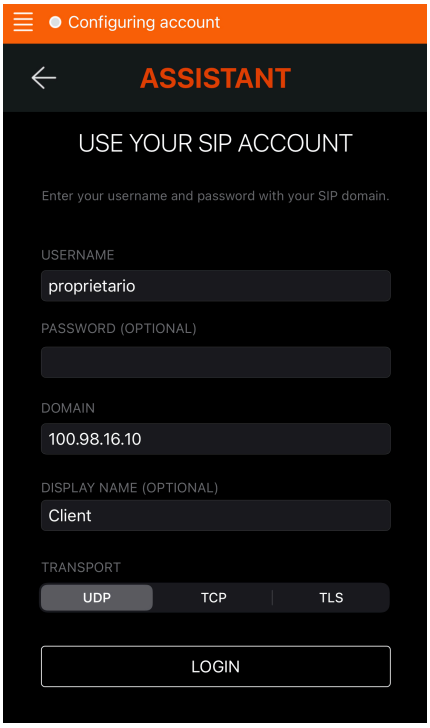
Fonte: O autor.

5 Resultados

“Há apenas um bem, o saber;
e apenas um mal, a ignorância.”
Sócrates

Após configurar todos os itens separadamente, foi preparado um ambiente de testes para validar o protótipo. Os testes iniciais foram focados no funcionamento do servidor VOIP usando o aplicativo Linphone para Adnroid e iOS. O aplicativo Linphone fornece uma interface gráfica para receber e realizar ligações através de um servidor SIP. É necessário adicionar as credenciais do cliente no servidor desejado bem como seu endereço IP para realizar a conexão inicial como ilustra a Figura 23.

Figura 23 – Tela de configuração do cliente SIP no aplicativo Linphone.



Fonte: Elaborada pelo autor.

Cada cliente possui suas credenciais. As credenciais são necessárias para que o Asterisk adicione ao arquivo `extension.conf` a rota de cada cliente. As rotas contém o informações como o endereço IP do cliente que deverá ser chamada no momento da ligação.

Com os clientes configurados foram realizados alguns testes de comunicação fazendo a ligação para o número `+551002`, utilizando ainda o aplicativo Linphone. O número `+551001` corresponde ao telefone do proprietário que está configurado dentro do Dialplan. Durante os testes foram notados:

- Comunicação estava clara e sem interferências entre ambas as partes;
- O tempo de resposta do servidor Asterisk foi menor que 1 segundo.

Com o servidor validado, foram realizados testes fazendo a ligação através do circuito criado utilizando o Raspberry-Pi e o *softphone* Twinkle recebendo a ligação no aplicativo criado usando a plataforma do Android Studio. Ao acionar o botão do circuito acoplado ao Raspberry-Pi (Figura 21), um *script* na linguagem Python dispara uma chamada para um *wrapper*, que é a instância de uma classe criada que encapsula o Twinkle e expõe somente as funcionalidades de iniciar e finalizar uma ligação, que é o precisamos. Esse *wrapper* possui duas funções públicas que fazem uso do Twinkle (Código 5.1): (i) `call_owner`, que faz a ligação para o proprietário, e (ii) `exit` que finaliza a ligação e encerra a chamada.

```
1 def call_owner(self):
2   self.twinkle_process.stdin.write(str.encode(f"call {self.owner_phone_number}\n
      "))
3   self.twinkle_process.stdin.flush()
4
5 def exit(self):
6   self.twinkle_process.stdin.write(str.encode("exit\n"))
7   self.twinkle_process.stdin.flush()
```

Listing 5.1 – *Wrapper* para iniciar e finalizar uma ligação no Raspberry-Pi

A chamada para o *wrapper* é realizada pela função `button_callback`, que foi configurada para ouvir o clique o evento `pull_up_down` no pino 10 (Código 4.1).

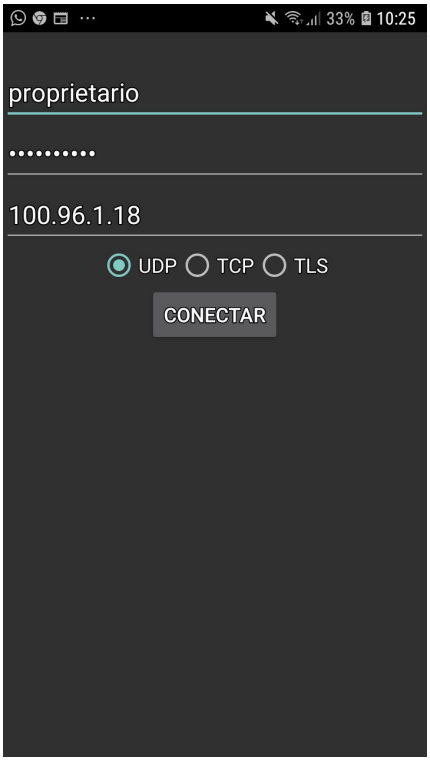
Por parte do aplicativo Android criado, a autenticação é feita usando a biblioteca Linphone. O Linphone fornece uma interface chamada `CoreListnerStub` onde é possível se conectar a um servidor VOIP e escutar eventos que o servidor envia, como por exemplo:

- **onAccountRegistrationStateChanged**: retorna se a autenticação no servidor VOIP foi feita com sucesso ou se teve falha;
- **onCallStateChanged**: método que retora os possíveis estados da chamada como por exemplo: *IncomingReceiver* que é chamado quando recebe uma ligação.

Como abordado na Seção 4.4.2, as telas do aplicativo criado para receber as ligações VOIP são: *login*, lista de visitantes e recebimento de chamadas. A tela inicial de *login* é para se autenticar no servidor VOIP (Figura 24). Ao se conectar o aplicativo traz uma lista que informa a data e hora em que o botão do Raspberry-Pi foi pressionado.

Quando o servidor VOIP envia uma chamada o método `onCallStateChanged` é chamado passando o estado de `IncomingReceiver` como parâmetro e então é aberto a tela de chamada dando a opção de atender ou recusar, como ilustra a Figura 25.

Figura 24 – Tela de login.



Fonte: Elaborada pelo autor.

Figura 25 – Tela de atender ligação.

Fonte: Elaborada pelo autor.

6 Conclusão

“As palavras fogem quando precisamos delas e sobram quando não pretendemos usá-las.”
Carlos Drummond de Andrade

Este trabalho justifica-se pela necessidade de se implementar sistemas de automação residencial que possibilite o monitoramento, recuperação de informação e tomada de decisão em residências de qualquer lugar do mundo, trazendo maior conforto e segurança aos moradores. Para isso, se buscou uma compreensão das tecnologias utilizadas por interfones que já possuem alguns recursos de ligação via central telefônica e outras tecnologias que poderiam agregar valor ou substituir os modelos atuais.

Inicialmente foram levantadas duas tecnologias possíveis de se utilizar para realizar a comunicação entre o interfone e um aparelho celular:

1. Integrar uma SDK Alexa da Amazon que engloba a funcionalidade de ligação telefônica para dispositivos Alexa;
2. Criar e configurar um servidor e dois clientes VOIP.

A primeira tentativa foi pesquisar a utilização da SDK Alexa da Amazon. A intenção de usar essa biblioteca se fez jus pelo crescimento da utilização dos dispositivos Alexa nas residências e por eles já virem com a opção de fazerem ligações (*built-in*) para outros dispositivos Alexa. Com isso, ao se instalar a biblioteca SDK em um sistema embarcado como o Raspberry-Pi, por exemplo, seria possível ligar para outro dispositivo Alexa que estivesse dentro de casa ou até mesmo para o aplicativo Alexa instalado no celular do proprietário. Entretanto, a partir de uma pesquisa feita sobre o tema, foi constatado que as ligações *built-in* só estão disponíveis para dispositivos Amazon Echo (COULTER, 2020).

Assim, este trabalho empreendeu uma análise da tecnologia VOIP e seus fundamentos e protocolos bem como sua aplicação técnica. Ao pesquisar sobre a tecnologia VOIP foram identificadas características que tornaram viáveis a criação de um protótipo que atendesse ao objetivo do trabalho, como:

- Portabilidade: existem bibliotecas que fornecem suporte e documentação para utilização de VOIP para diferentes sistemas embarcados (Raspberry-Pi, *smart phones*, etc);
- Modularidade: embora o sistema telefônico possa parecer misterioso, um sistema VOIP moderno não precisa ser complexo. O *disign* modular do Asterisk permite habilitar somente as funcionalidade que você necessita;
- Documentação: material amplamente difundido na comunidade *open source*, possuindo documentações e exemplos de implementações.

Quatro resultados foram obtidos neste trabalho. O primeiro resultado refere-se a um ao servidor VOIP Asterisk que foi desenvolvido para a comunicação entre o interfone e um telefone celular. O segundo resultado refere-se a instalação e configuração do circuito no Raspberry-Pi fazendo com que funcione como um interfone e consiga realizar uma chamada VOIP. O terceiro resultado trata-se da criação de um aplicativo que recebe chamadas VOIP e possua uma lista com as informações de data e hora em que o interfone de sua casa tocou. O quarto resultado refere-se à validação da integração entre as três primeiras partes.

Dentre as atividades desenvolvidas o principal desafio foi encontrar e configurar um *softphone* para o Raspberry-Pi que fornecesse suporte via interface de linha de comando (CLI) para disparar uma chamada assim que o usuário pressionasse o botão do circuito GPIO.

Por fim, embora o trabalho tenha cunho experimental e didático, o resultado obtido na comunicação entre o Raspberry-Pi e o aplicativo através do servidor VOIP demonstrou ser eficiente e uma alternativa de automação residencial para interfone entre diversas outras que já existem no mercado.

6.1 Trabalhos futuros

Os resultados deste trabalho sugerem algumas direções promissoras de trabalho futuro:

- Adição da funcionalidade de vídeo na chamada;
- Criação da opção de abrir o portão via aplicativo;
- Novos estudos sobre a integração com assistentes virtuais inteligentes como Alexa (AMAZON, 2020a), Siri (APPLE, 2021), Google *Assistant* (DEVELOPERS, 2021b).

Referências

ACADEMY, D. S. *Capítulo 1 – Deep Learning e a Tempestade Perfeita*. 2018. Disponível em: <http://deeplearningbook.com.br/deep-learning-a-tempestade-perfeita/>. Citado nas páginas 18 e 19.

ACADEMY, D. S. *Capítulo 76 – O Que é BERT (Bidirectional Encoder Representations from Transformers)?* 2018. Disponível em: <http://deeplearningbook.com.br/o-que-e-bert-bidirectional-encoder-representations-from-transformers/>. Citado na página 19.

AGRAWAL, D. V. S. A survey on internet of things. *Abakós, Belo Horizonte*, v. 1, n. 2, p. 77–95, 2013. Citado na página 15.

AMAZON. *Get Started with the Alexa Skills Kit*. 2020. Disponível em: <https://developer.amazon.com/en-US/alexa/alexa-skills-kit/start>. Citado nas páginas 20, 22 e 48.

AMAZON. *O que são as skills da Alexa?* 2020. Disponível em: <https://www.amazon.com.br/gp/help/customer/display.html?nodeId=GG3RZLAA3RH83JAA>. Citado nas páginas 17 e 20.

AMAZON. *What is the Alexa Voice Service?* 2020. Disponível em: <https://developer.amazon.com/en-US/docs/alexa/alexa-voice-service/get-started-with-alexa-voice-service.html>. Citado na página 21.

APPLE. *The speakers of the house.* 2021. Disponível em: <https://www.apple.com/homepod/>. Citado na página 48.

AURESIDE. *Automação Residencial: demanda na Construção Civil*. 2016. Disponível em: <www.aureside.org.br>. Citado nas páginas 12 e 13.

BAMPI, A. V. *PROTÓTIPO DE INTERFONE SEM FIO COM ÁUDIO E VÍDEO*. 2018. 104 p. Dissertação (CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO) — UNIVERSIDADE REGIONAL DE BLUMENAU, Blumenal, SC, Brasil, 2018. Citado na página 13.

BLYNK. *We make Internet of Things simple for you*. 2021. Disponível em: <https://blynk.io>. Citado na página 22.

BRYANT, B. K. F. R. *Asterisk Versions*. 2021. Disponível em: <https://wiki.asterisk.org/wiki/display/AST/Asterisk+Versions>. Citado na página 25.

BUSINES, F. *Market Research Report*. 2019. Disponível em: <https://www.fortunebusinessinsights.com/industry-reports/home-automation-market-100074>. Citado nas páginas 12 e 13.

BUSINES, F. *Race Ahead of Competition, Create a Strong Brand Image*. 2019. Disponível em: <https://www.fortunebusinessinsights.com/>. Citado na página 12.

CAMPOS, G. *Introdução a Machine Learning e TensorFlow*. 2017. Disponível em: <https://www.slideshare.net/GuilhermeCampos77/introduo-a-machine-learning-e-tensorflow>. Citado na página 18.

COULTER, S. *Calling feature test through C++ Alexa SampleApp*. 2020. Disponível em: <https://github.com/alexa/avs-device-sdk/issues/1681>. Citado na página 47.

CRITELLI, A. *How to configure a SIP endpoint for intra-office calling*. 2020. Disponível em: <https://www.redhat.com/sysadmin/sip-endpoint>. Citado nas páginas 34, 36 e 37.

CRITELLI, A. *How to install Asterisk on Linux*. 2020. Disponível em: <https://www.redhat.com/sysadmin/install-asterisk-linux>. Citado nas páginas 32 e 33.

CRITELLI, A. *An introduction to Asterisk*. 2020. Disponível em: <https://www.redhat.com/sysadmin/introduction-asterisk>. Citado nas páginas 27 e 35.

CRITELLI, A. *An introduction to Asterisk*. 2020. Disponível em: <https://www.redhat.com/sysadmin/introduction-asterisk>. Citado na página 28.

CRITELLI, A. *An introduction to VoIP for sysadmins*. 2020. Disponível em: <https://www.redhat.com/sysadmin/introduction-voip>. Citado nas páginas 17 e 40.

DAMACENO, R. O. V. S. S. InteligÊncia artificial: Uma breve abordagem sobre seu conceito real e o conhecimento popular. *Caderno de Graduação - Ciências Exatas e Tecnológicas - Universidade Tiradentes - UNIT*, v. 5, n. 1, p. 11–16, 2018. Disponível em: <https://periodicos.set.edu.br/cadernoexatas/article/view/5729>. Citado na página 18.

DECORTIPS. *Arquitetura domótica: saiba mais! - Decor Tips*. 2021. Disponível em: <https://platform.ifttt.com/docs>. Citado na página 15.

DENG, Y. L. L. *Deep Learning in Natural Language Processing*. 152 Beach Road, Singapore: Springer Nature Singapure, 2018. 1-3 p. Citado na página 19.

DEVELOPERS, G. *OpenVPN Connect – Fast Safe SSL VPN Client*. 2018. Disponível em: <https://play.google.com/store/apps/details?id=net.openvpn.openvpn>. Citado na página 43.

DEVELOPERS, G. *Android Studio*. 2021. Disponível em: <https://developer.android.com/>. Citado nas páginas 25 e 39.

DEVELOPERS, G. *Available on Speakers*. 2021. Disponível em: <https://assistant.google.com/platforms/speakers/>. Citado na página 48.

DEVELOPERS, G. *Desenvolver apps Android com o Kotlin*. 2021. Disponível em: <https://developer.android.com/kotlin>. Citado na página 39.

DEVELOPERS, G. *Linphone - open source VOIP project*. 2021. Disponível em: <https://www.linphone.org/technical-corner/linphone>. Citado na página 39.

ELLINGWOOD, J. *Como configurar um servidor OpenVPN no Ubuntu 18.04*. 2021. Disponível em: <https://www.digitalocean.com/community/tutorials/how-to-set-up-an-openvpn-server-on-ubuntu-18-04-pt>. Citado na página 42.

EUZÉBIO, M. V. . M. E. r. droidlar- automação residencial através do celular android. sistemas de telecomunicações. *Instituto Federal de Santa Catarina. São José*, Santa Catarina, v. 28, n. 5, 2011. Citado na página 12.

FOUNDATION, O. *About Node.js®*. 2021. Disponível em: <https://nodejs.org/en/>. Citado na página 20.

FOUNDATION, R. P. *Raspberry Pi 4*. 2021. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/>. Citado na página 16.

FREEZVON. *How to install and configure Twinkle softphone for Linux system*. 2017. Disponível em: <https://freezvon.com/services/twinkle>. Citado na página 38.

GROSS, F. D. *Implementando uma Solução VoIP Baseada em Asterisk com Alta-disponibilidade e Balanceamento de Carga*. 2009. 8 p. Dissertação (Trabalho de conclusão de curso) — Universidade Federal de Santa Catarina, SC, Brasil, 2009. Citado nas páginas 17, 26 e 27.

GUSMÃO, J. B. L. Sistema de acesso remoto via gms para interfones. *Centro Universitário de Brasília - CEUB*, Brasília, n. 5, 2012. Citado na página 13.

HOME, S. of the S. *Control Networks*. 2015. Disponível em: <http://www.icontrol.com/wp-content/uploads/2015/06/Smart_Home_Report_2015.pdf>. Citado na página 12.

HQ, R. P. *Using a push button with Raspberry Pi GPIO*. 2018. Disponível em: <https://raspberrypihq.com/use-a-push-button-with-raspberry-pi-gpio/>. Citado na página 41.

HUTCHINSON, P. *Raspberry Pi 4, 3 B+, Pi 3, Pi 2, B+, A+ Comparison Chart*. 2019. Disponível em: <https://www.element14.com/community/docs/DOC-68090/l/raspberry-pi-4-3-b-pi-3-pi-2-b-a-comparison-chart>. Citado na página 16.

IBGE. 2010. Disponível em: <www.ibge.gov.br>. Citado na página 12.

IFTT. *Documentation*. 2021. Disponível em: <https://platform.ifttt.com/docs>. Citado na página 22.

INC, R. *SCan't Get to the Door? Now Your Ring Doorbell Answers for You With Smart Responses*. 2021. Disponível em: <https://blog.ring.com/2021/02/10/now-your-ring-video-doorbell-answers-the-door-for-you-with-smart-responses/>. Citado na página 24.

JETHA, H. *How to Set Up SSH Keys on Ubuntu 18.04*. 2018. Disponível em: <https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys-on-ubuntu-1804>. Citado na página 42.

JORDAN, M. *Dialplan*. 2015. Disponível em: <https://wiki.asterisk.org/wiki/display/AST/Dialplan>. Citado na página 31.

LLP, P. *Host, run, and code Python in the cloud!* 2021. Disponível em: <https://www.pythonanywhere.com>. Citado na página 25.

MALCOM, M. J. *Installing Asterisk From Source*. 2021. Disponível em: <https://wiki.asterisk.org/wiki/display/AST/Installing+Asterisk+From+Source>. Citado nas páginas 27, 28 e 32.

MEGGELEN RUSSELL BRYANT, L. M. J. V. *Asterisk: The Definitive Guide*. USA: O'Reilly Media, Inc., 2019. Citado na página 31.

MICHAELIS. *Dicionário Brasileiro da Língua Portuguesa*. 2020. Disponível em: <http://michaelis.uol.com.br/moderno-portugues/busca/portugues-brasileiro/artificial>. Citado na página 17.

NETO, L. D. *PROTÓTIPO DE AUTOMAÇÃO RESIDENCIAL UTILIZANDO UMA ASSISTENTE DE VOZ*. 2018. 61 p. Dissertação (CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO) — UNIVERSIDADE REGIONAL DE BLUMENAU, Blumenal, SC, Brasil, 2018. Citado nas páginas 22, 23 e 24.

NEWTON, R. *Asterisk Architecture, The Big Picture*. 2014. Disponível em: <https://wiki.asterisk.org/wiki/display/AST/Asterisk+Architecture2C+The+Big+Picture>. Citado nas páginas 28, 29 e 30.

NEWTON, R. *Channels*. 2014. Disponível em: <https://wiki.asterisk.org/wiki/display/AST/Channels>. Citado nas páginas 30 e 31.

OPENVPN. *Secure networking, with a name you trust.* 2021. Disponível em: <https://openvpn.net>. Citado na página 42.

ORACLE. *Oracle Java®*. 2021. Disponível em: <https://www.oracle.com/br/java/>. Citado na página 20.

ORG, R. *Raspberry Pi 4*. 2021. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/>. Citado na página 25.

ORGANIZATION, P. *The Python Language Reference*. 2021. Disponível em: <https://docs.python.org/3/reference/>. Citado na página 20.

RASHID, T. *Make Your Own Neural Network*. North Charleston, SC, United States: Independent Publishing, 2016. 41-42 p. Citado na página 18.

RING. *Segurança Inteligente*. 2021. Disponível em: <https://latam-es.ring.com>. Citado na página 24.

SEGURANCA, A. *Interfone sem fio: o que é, e como funciona?* 2019. Disponível em: <https://blog.segurancaalfa.com.br/interfone-sem-fio-como-funciona/?fbclid=IwAR380jbKTGg_vWUYhmC-JvO0IM6Pxp4DYYjGNSxhAwZcO308chz9ZTKDcgg>. Citado na página 13.

SGARBI, F. T. J. A. Domótica inteligente: Automação residencial baseada em comportamento. *Centro Universitário da FEI – UniFEI*, n. 5, 2006. Citado na página 15.

SILVA, V. L. D. M. Caio Alexandre da. AutomaÇÃo residencial com inteligencia artificial. *Revista Inovação*, n. 5, p. 22, 2018. Citado nas páginas 22 e 23.

SQUARE, I. *Retrofit - A type-safe HTTP client for Android and Java*. 2021. Disponível em: <https://square.github.io/retrofit/>. Citado na página 40.

TRILLA, A. Natural language processing techniques in text-to-speech synthesis and automatic speech recognition. *Departament de Tecnologies Me'dia Enginyeria i Arquitectura La Salle (Universitat Ramon Llull)*, Barcelona, Spain, v. 1, n. 1, 2009. Disponível em: <http://atrilla.net/data/files/micnlp09.pdf>. Citado na página 19.

TWINKLE. *Twinkle*. 2017. Disponível em: <http://twinkle.dolezel.info>. Citado na página 38.

UBUNTU. *Download Ubuntu Desktop*. 2021. Disponível em: <https://ubuntu.com/download/desktop>. Citado na página 25.

WILLIAN SANTOS RENATO, C. D. L. J. J. 2019. 46 p. Dissertação (Mestrado) — Repositório Institucional da Universidade Tecnológica Federal do Paraná (RIUT), Ponta Grossa, Paraná, Brasil, Title = SISTEMA DE AUTOMATIZAÇÃO RESIDENCIAL DE BAIXO CUSTO CONTROLADO PELO MICROCONTROLADOR ESP32 E MONITORADO VIA SMARTPHONE, 2019. Disponível em: <http://repositorio.utfpr.edu.br/jspui/handle/1/16960>. Citado na página 13.

ZHAO JAYANAND JEGATHEESAN, S. C. L. C. W. Exploring iot application using raspberry pi. *CComputer Networks and Applications*, n. 5, 2015. Citado na página 16.