Projeto de um sistema de reencaminhamento de chamadas de interfone para dispositivos móveis

Vinícius Faustino Silva

Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica
Graduação em Engenharia de Computação

Uberlândia
2018

Vinícius Faustino Silva

Projeto de um sistema de reencaminhamento de chamadas de interfone para dispositivos móveis

Trabalho de Conclusão de Curso apresentado a Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Engenheiro de Computação.

Área de concentração: Engenharia de Computação

Orientador: Kil Jin Brandini Park

Uberlândia
2018

Resumo

A criação deste projeto foi baseada no desenvolvimento de um sistema que faça o uso de diversas tecnologias e, consequentemente, para sua concretização foi desenvolvido uma aplicação mobile destinada ao sistema operacional Android e outra que utiliza o hardware de um microprocessador com software embarcado. O sistema tem como propósito estabelecer uma comunicação entre um interfone residencial e o smartphone do morador. O microprocessador utilizado foi um Raspberry Pi 3 model B programado com um algoritmo em linguagem Python e a aplicação mobile construída com a ferramenta Android Studio, utilizando a linguagem Java. Ao fazer o acionamento do interfone, o dispositivo embarcado criará um meio de comunicação com o smartphone por meio de um banco de dados online, o Firebase. Ao término do projeto, com sua implementação concretizada, uma limitação foi observada. Ao fazer a captura da voz de quem está de frente ao interfone, é utilizado uma biblioteca de reconhecimento de voz e, caso haja muito barulho ambiente, o reconhecimento pode apresentar falhas.

Palavras-chave: Automação residencial; Hardware de Baixo Custo; Raspberry Pi; Android; Banco de dados; Integração web.

Abstract

The creation of this project was based on the development of a system that makes use of several technologies and, consequently, for its accomplishment a mobile application was developed for the Android operating system and another one that uses the hardware of a microprocessor with embedded software. The purpose of the system is to establish a communication between a residential intercom and the resident's smartphone. The microprocessor used was a Raspberry Pi 3 model B programmed with a Python language algorithm and the mobile application built with the Android Studio tool, using the Java language. When launching the intercom, the embedded device will create a means of communication with the smartphone through an online database, Firebase. At the end of the project, with its implementation implemented, a limitation was observed. When capturing the voice of the person standing in front of the intercom, a voice recognition library is used, and if there is too much ambient noise, recognition may fail.

Keywords: Home Automation; Low Cost Hardware; Raspberry Pi; Android; Database; Web Integration.

Lista de ilustrações

Lista de abreviaturas e siglas

API	Application Programming Interface
APP	Application
CPU	Central Process Unit
GPIO	General Purpose Input Output
IDE	Integrated Development Environment
IoT	Internet of Things
OHA	Open Handset Alliance
OSHW	Open Source Hardware
POO	Programação Orientada a Objetos
RAM	Random Access Memory
ROM	Read only memory
Sdk	Software development kit
SHA-1	Secure Hash Algorithm 1
SMS	Short Message Service
SO	Sistema Operacional
UI	User Interface
URL	Uniform Resource Locator
USB	Universal Serial Bus
XML	Extensible Markup Language

Sumário

Capítulo 1

Introdução

Com a evolução da automação na indústria foi construído uma padronização, uniformidade e uma produção mais flexível. Com essa evolução vinda da indústria e, buscando uma qualidade de vida melhor para as pessoas, a automação passou a fazer parte também do ambiente doméstico (FERREIRA, 2010).

A automação residencial foi, no começo, entendida como uma representação de status e modernidade, porém, com o passar dos tempos, passou a ser melhor definida por conta de apresentar conforto, economia e segurança para quem faz-se uso desta evolução tecnológica (CABRAL; CAMPOS, 2008).

O termo domótica é sinônimo de automação residencial pois, assim como a automação, ela traz a idéia da utilização da tecnologia para fazer o controle de recursos habitacionais. A palavra domótica nasceu da junção da palavra "domus", que é originária do Latim e significa casa, com a palavra "robótica", que está ligada a ação de automação.

Considerando a inserção intensa de tecnologia em praticamente todas as coisas, os eletrodomésticos são alvos certeiros. Com o surgimento das tecnologias denominadas IoT, do inglês, internet of things, que pode ser traduzido como internet das coisas, a automação residencial evoluiu absurdamente, como exemplo, é possível fazer desde o controle da luminosidade das lâmpadas até monitorar remotamente uma residência.

Como a disseminação deste tipo de tecnologia foi tão intensa, no Brasil, surgiram diversas empresas que trazem como propósito fazer a automação residencial. Porém para se adquirir esse produto é necessário um custo maior para o investimento. Como resultado tem-se um custo muito alto em relação ao benefício auferido.

Devido ao alto custo para produção de hardwares específicos para automação, foi gerado uma procura por outros caminhos para se conseguir os mesmo resultados. E então, para gerar um menor custo na produção desses hardwares, surgiu a possibilidade de usar um novo conceito de hardware chamado de open-source hardware (OSHW), ou hardware livre.

Open Source Hardware é um termo destinado a dispositivos, artefatos físicos, mecânicos ou elétricos, onde seus dados do projeto sejam visíveis e compartilhados a qualquer

pessoa, possibilitando assim a modificação, distribuição e utilização desses dispositivos (TAPR, 2007).

Atualmente com a concepção de open source hardware bem difundida vem surgindo dispositivos cada vez mais potentes e de baixo custo, sendo os mais populares a Raspberry e Arduino.

A maior rede de computadores existente não poderia ficar de fora do contexto de automação residencial. A internet vem se integrando a soluções diversas para diversos tipos de aplicações e junto a automação residencial traz um nível maior de conforto e segurança para os usuários.

Tecnologias como câmeras que podem ser controladas remotamente, por meio da rede de computadores fazendo que o usuário consiga imagens em tempo real, além de alarmes que quando disparados, são capazes de avisar ao usuário e ainda acionar a polícia caso esteja configurado. Com isso em mente, a automação residencial em conjunto com a internet, além de aumentar significativamente a comodidade oferecida, traz consigo mais segurança, o que nos dias de hoje é excepcionalmente necessário.

Junto da internet e o open source hardware, outra tecnologia que eclodiu de forma exorbitante foram os smartphones.

Os smartphones evoluíram de uma forma que conseguem oferecer as mesmas funcionalidades de um computador, consequentemente fazendo algo de suma importância, conectar as pessoas a internet sem o uso do computador. Com a evolução constante dos smartphones, com a inserção de hardwares melhores e softwares melhores, os telefone celulares que anteriormente eram apenas para efetuar ligações, se transformaram em dispositivos poderosos muito importantes do dia a dia das pessoas (RUWAIDA; MINKKINEN, 2013).

Com o smartphone podendo então praticamente substituir o computador, o "controle remoto" da residência não podia ficar fora do aparelho. E foi pensando nisso que as empresas resolverem investir fortemente em soluções integradas com o smartphone. Portanto é possível encontrar no mercado diversas soluções que fazem do smartphone sua central de comandos, tendo em mente poder fazer desde o controle de eletrodomésticos pelo aparelho até o monitoramento via câmeras na palma da mão.

Quando se fala sobre smartphones o que vem em mente são as duas maiores empresas em desenvolvimento mobile: Apple e Google. No caso da Apple com seu grande produto de sucesso, o Iphone, que utiliza o sistema IOS que vem sendo desenvolvido a anos.

Neste projeto será utilizado um sistema Android, da gigantesca Google. Inicialmente o Android não pertencia ao Google, em 2003 foi fundada a empresa com o nome Android que tinha como foco as câmeras digitais, buscando trazer um sistema operacional inteligente para elas. Mas a empresa percebeu que era viável fazer o desenvolvimento do produto para celulares e em 2005 foi então comprada pela Google (TECMUNDO, 2017).

O sistema é baseado em linux e faz uso de uma interface de usuário conceituada na manipulação direta. Neste projeto foi desenvolvido um aplicativo nativo para este sis-

tema operacional e utilizando juntamente com o Firebase, também da gigante Google. O Firebase é uma ferramenta do Google para desenvolvimento de aplicativos mobile e da web. Dentre diversas ferramentas inclusas no Firebase como storage, hosting, authentication entre outras, foi utilizada o banco de dados em tempo real (Realtime Database). O sistema de banco de dados em tempo real fornece um banco de dados não relacional, NoSQL, que tem como principal funcionalidade o fato de poder adicionar escutas ao banco e assim o próprio Firebase informa quando algo acontecer no banco, portanto não tendo a necessidade de ficar fazendo constante consultas buscando por novas modificações.

A parte desenvolvida em hardware será utilizando o minicomputador Raspberry Pi, que por suas características de hardware permite fazer a instalação do sistema operacional adequado, dentro de vários disponibilizados via open source. Este pequeno componente pesando apenas 45g com capacidade de funções fantásticas foi lançado em 2012, e por causa do seu tamanho e peso ela se torna perfeita para utilização em projeto de automação residencial, já que pode ser instalada em diversos e pequenos lugares como dentro de uma caixa elétrica, ou ainda pode assumir a função de termostato instalado em uma parede (DENNIS, 2013).

Mais especificamente neste projeto foi utilizado o modelo Raspberry Pi 3 Model B v1.2 que, fazendo uso de um algoritmo desenvolvido em Python, que é uma linguagem de alto nível, interpretada e orientada a objetos, irá trocar mensagens com o banco online, capturar áudio do ambiente consequentemente trabalhando como o interfone.

1.1 Objetivo

O objeto deste projeto é desenvolver um sistema que faça uso de diversas tecnologias voltadas para hardware e software, utilizando assim uma Raspberry Pi 3 Model B com um sistema operacional embarcado e um software construído na linguagem Python, utilização da internet como meio de comunicação, um banco de dados em tempo real e desenvolvimento de uma aplicação Android.

A criação deste sistema tende a substituir o interfone convencional e trará um novo produto, um interfone que possa ser atendido por meio do smartphone. O produto funcionará da seguinte forma: quando o visitante tocar o interfone da residência será disparado um evento no microprocessador, que será uma Raspberry Pi 3 Model B, e ao ser feita essa captura do acionamento do interfone será armazenado um registro no banco de dados Realtime Database do Firebase e assim, o aplicativo Android que estará rodando no smartphone do morador da residência irá ver esse novo registro e disparará uma notificação avisando que alguém está tocando o interfone da residência. Caso o morador queira atender a esta notificação, ao clicar na mesma a aplicação mobile irá passar para uma tela onde é possível fazer o atendimento da chamada do interfone e, caso atendida, o sistema entrará em modo de atendimento, e então haverá a transmissão de informações

entre as duas pontas, isto é, o interfone e o smartphone do morador. O morador terá na interface do aplicativo uma série de respostas pré definidas, como por exemplo “Quem é?” ou “Só um instante por favor”, e poderá escolher quais mandar para o interfone executar. Selecionando uma resposta, ela será enviada ao banco e assim o microprocessador recuperará a mensagem e irá exibi-la em forma de áudio para o visitante. Já do outro lado, quando o visitante falar algo, isso será transformado em texto para envio ao banco de dados e assim o aplicativo Android fará a captação da resposta e apresentará ao morador em forma de texto. Portanto o morador poderá “atender” ao visitante estando ou não em casa, ou seja, saberá quem foi à sua residência mesmo não estando nela. Além disso terá na tela principal do aplicativo a visão de todas as conversações realizadas, tendo então um histórico de quem foi a sua residência.

Portanto para concretização deste sistema foi feito toda a configuração da Raspberry Pi, desde instalação do sistema operacional até ajustes de configurações de áudio. Foi desenvolvido um software em Python para controle das tarefas da Raspberry Pi, fazendo o desenvolvimento desde a captação de áudio, acionamento do botão do interfone, emissão de áudio, a conexão e conversação com o banco online, desenvolvido um aplicativo para o sistema operacional Android, que é a interface entre o usuário morador e o usuário visitante, a parte de front-end, criação dos layouts e back-end, comunicação com os elementos gráficos, conexão com banco, toda lógica de programação. Então, foi definido etapas de comunicação entre o smartphone e a Raspberry Pi, fazendo com que toda essa comunicação ocorra via internet por meio da base de dados, permitindo ao usuário ter acesso às chamadas do interfone de qualquer lugar, apenas tendo acesso a internet.

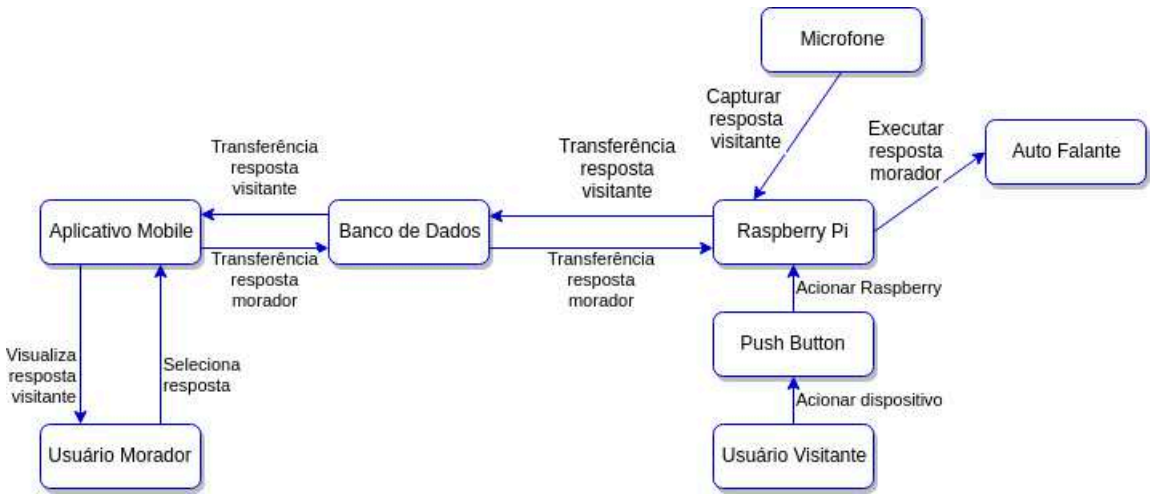


Figura 1 – Fluxograma de informações.

A figura 1 apresenta todo o fluxo de informações que transitam de acordo com o modelo de design escolhido. Com ambos os sistemas já iniciados, a raspberry fica aguardando pelo acionamento do push button que representa o botão do interfone e o aplicativo mobile fica

ouvindo se algum novo nó é adicionado ao banco de dados em background. Quando um visitante faz o acionamento do push button a Raspberry cria um novo nó no banco, o qual o aplicativo mobile "ouve" e então gera a notificação para o usuário. Este, atendendo ao chamado, pode enviar uma mensagem para Raspberry executar para o visitante e então é capturado o áudio de resposta do visitante e enviado ao aplicativo por meio do banco de dados.

1.2 Justificativa/Relevância

Com a conclusão do projeto será possível gerar alguns benefícios para o usuário que decidiu adquirir o sistema. Dentre eles, pode-se citar:

- ❏ Utilização de um hardware de baixo custo, pelo fato de poder ser utilizado em diversos projetos e a flexibilidade que entrega ao usuário, trazendo assim uma economia maior ao adquirir este produto em relação a outros de hardware específico.
- ❏ Open Source Hardware, contribuição para continuação de projetos de código aberto.
- ❏ Conforto, sistema permite atender ao interfone sem necessidade de movimentação por parte do morador.
- ❏ Segurança, permite-se ter um nível maior de controle de quem vai a residência, por meio de um histórico das chamadas e a possibilidade de atendimento do interfone mesmo não estando na residência.
- ❏ O sistema tem como intuito fazer uso da maior rede de computadores existente, a internet. Como atualmente praticamente ninguém fica sem acesso a internet e a popularização da automação residencial vem crescendo fortemente, este sistema se adequa perfeitamente a estes temas atuais.

O conhecimento adquirido neste projeto será extremamente útil no desenvolver de outros sistemas, em oportunidades de trabalho e direcionamento na carreira acadêmica. A aprendizagem de tecnologias open source, sistemas embarcados, desenvolvimento em linguagem orientada a objetos e interpretada, aplicação mobile Android e gerenciamento de banco de dados, será com certeza de suma importância para demais projetos da engenharia.

Capítulo 2

Embasamento Teórico

2.1 Sistemas Embarcados

Sistemas embarcados tem como propósito servir a uma atividade específica, fazendo com que seja possível desenvolver um sistema próprio para tal atividade fazendo deste um sistema muito bem ajustado para determinadas funções, conseguindo resultados melhores do que quando se utiliza um sistema não otimizado para uma tarefa. Com este propósito, sistemas embarcados traz consigo um forte acoplamento do hardware e software por ele utilizados.

Computadores são denominados sistemas de propósito geral pois, são destinados a realizar diversos tipos de tarefas, jogos, trabalho, entretenimento, desenvolvimento, portanto um computador deve estar apto a realizar diversas tarefas para cumprir com seu objetivo. Já os sistemas embarcados vem com a intenção de realizar conjuntos de tarefas bem definidos, e consequentemente com requisitos de funcionamento bem específicos.

Os sistemas embarcados vem crescendo fortemente e estão cada vez mais presente no nosso dia a dia em diversos tipos de aparelhos como, por exemplo, smartphone, eletrodomésticas, automóveis, etc. Estes sistemas vêm se tornando cada vez mais baratos e acessíveis, e como tem um custo energético baixo e um tamanho físico relativamente pequeno, vem agradando seus usuários.

Diferentemente de um computador que possui sua capacidade de armazenamento ajustável a necessidade do usuário, sistemas embarcados possuem certas limitações, ao fazer a utilização de sistemas embarcados, seus dados e sistemas são armazenados normalmente em memória FLASH ou ROM.

Para conseguir gerar os melhores resultados para uma determinada aplicação, estes sistemas embarcados fazem o uso de diversas arquiteturas. Há sistemas construídos para aplicações mais simples, como sensoriamento remoto ou controle de máquinas de lavar, que fazem o uso de uma arquitetura de 8 bits como, por exemplo, o 8051. Além dos sistemas de 16 bits, há os com a arquitetura de 32 bits, que já são sistema bem mais complexos, onde sua programação já é mais feita em alto nível, e estes são os que mais se

assemelham aos computadores pessoais (OTONI, 2013).

2.2 Programação Orientada a Objetos

Por causa do desenvolvimento de software ser bastante extenso, cada linguagem tem a possibilidade de seguir um paradigma diferente. Dos diversos paradigmas existentes, os que mais se difundiram foram os conhecidos como programação orientada a objetos e a programação estruturada.

Para entender um pouco melhor a diferença entre POO e programação estruturada é válido observar o que acontece com os dados globais. Na programação estruturada os métodos, também denominados funções, são aplicados globalmente na aplicação, o que não ocorre na programação orientada a objetos, onde cada objeto tem seus próprios métodos e por sua vez, estes são aplicados a seus dados.

Uma linguagem conhecida por praticamente todos que fazem parte desse mundo tecnológico, é a linguagem C, a principal linguagem estruturada. Está se trata de uma linguagem que se adequa muito bem como linguagem de baixo nível, muito utilizada para programação de sistemas embarcados e sistemas que necessitam de um conhecimento mais profundo do hardware.

Já há algum tempo, a programação orientada a objetos vem sendo cada vez mais difundida entre os desenvolvedores, por apresentar vários pontos relevantes e pelo fato de o desenvolvedor não ter a necessidade de se preocupar com o desempenho da aplicação, por causa dos poderosos processadores que temos hoje em dia, que proporcionam um desempenho fantástico. Outro ponto de suma importância que facilitou a difusão da POO, é o fato dela buscar uma representação do mundo real de forma mais fiel, com a utilização de objetos e da reutilização de código.

Dentre os vários fatores que caracterizam a programação orientada a objetos, há quatro que são as bases desse paradigma: abstração, encapsulamento, herança e polimorfismo.

2.2.1 Abstração

A abstração pode ser dividida em três pontos importantes, identidade, propriedades e métodos.

Cada objeto criado no sistema é único, e esta unicidade é proporcionada pela introdução de um identificador para cada objeto. Este identificador permite distinguir um objeto de outro sem a necessidade de verificar suas propriedades ou métodos. Portanto cada identificador é único dentro de seu pacote.

Como o papel do objeto é tentar representar os objetos do mundo real, este elemento da POO, assim como na vida real, precisa ter aspectos que lhe represente. Como exemplo, pode-se citar uma pessoa, que tem como características que a definem ter um tamanho, sexo, idade, cpf. Estas característica na programação são chamada de propriedades.

E por fim, um objeto pode apresentar seus métodos, que são ações que o objeto pode executar que, por exemplo, em um objeto que represente uma pessoa, poderia ser “falar()”, “dormir()” ou em um cachorro, “latir()”.

2.2.2 Encapsulamento

O encapsulamento é uma das mais importantes características que definem a programação orientada a objetos pois ele permite fazer a adição de segurança ao programa, podendo esconder as propriedades de um objeto.

Para evitar o acesso direto às propriedades da aplicação, a maior parte das linguagens orientadas a objetos implementam métodos especiais que fazem as alterações das propriedades (setters) e métodos que fazem a devolução do valor das mesmas (getters).

2.2.3 Herança

A herança é o elemento responsável pela potencialização da POO proporcionando o reaproveitamento de código, que gera uma melhoria no tempo de produção e diminuição no número de linhas. Assim como na vida real que um filho pode herdar características de seu pai, na POO, um objeto pode herdar os métodos e propriedades de outro. Podendo então haver heranças múltiplas, onde há o encadeamento de heranças.

2.2.4 Polimorfismo

Este elemento trabalha junto a herança e é necessário quando um método do objeto pai precisa ser alterado no objeto filho. Por exemplo, um objeto denominado animal tem um método chamada “emitirSom()” que é comum a todos os animais, porém cada animal emite um som diferente, então quando um cachorro herdar a classe animal, será necessário reescrever o método de acordo com as características específicas do cachorro. O mesmo aconteceria com um gato, herdaria a classe animal, porém faria as alterações no método “emitirSom()”.

2.3 Android

Desenvolvido buscando atingir dispositivos móveis, o Android, é o sistema operacional mais conhecido do mundo. Este sistema é capaz de gerenciar diversas tarefas, e cada uma com finalidades totalmente diferentes uma das outras como, por exemplo, permitir a instalação de aplicações que possibilitam personalizar um aparelho, reproduzir músicas, tirar fotos, ter acesso a internet, além de manter as funções básicas de um celular, como fazer ligações e envio de SMS por rede de telefonia. O Android passou a dominar uma grande gama de aparelhos, além dos celulares, como tablets, TVs e muito mais. Diferente

do que muitos pensam, o Android surgiu sendo desenvolvido pela empresa Android Inc e somente em julho de 2005 a gigante Google fez a compra do sistema.

Quando se iniciou o desenvolvimento do Android, já com a empresa pertencendo a Google, buscou desenvolver um sistema baseado em um sistema operacional similar ao Linux mas com um grande diferencial: a Google pretendia fazer esse desenvolvimento voltado para aparelhos móveis. O desenvolvimento foi com a utilização da linguagem Java e atualmente diversas empresas montaram um grupo conhecido como OHA (Open Handset Alliance). Este grupo de empresas tem como objetivo desenvolver e aumentar o número de aplicativos que são baseados no Android. São sistemas operacionais desenvolvido a partir do Android e o objetivo disso é fazê-lo realizar os mais variados tipos de tarefas (GOMES; FERNANDES; FERREIRA, 2012).

O Android se tornou tão popular e cresceu tanto que, de acordo com a figura 2, é possível concluir que passou a ser o sistema operacional mais utilizado no mundo. Em março de 2017 o sistema chegou a esta posição ultrapassando o sistema da Microsoft, o Windows. Nesta época o sistema operacional da Apple, o IOS, apresentava utilização que atingia a marca de aproximadamente 14% enquanto que o Android alcançou em mais de uma década aproximadamente 40%.

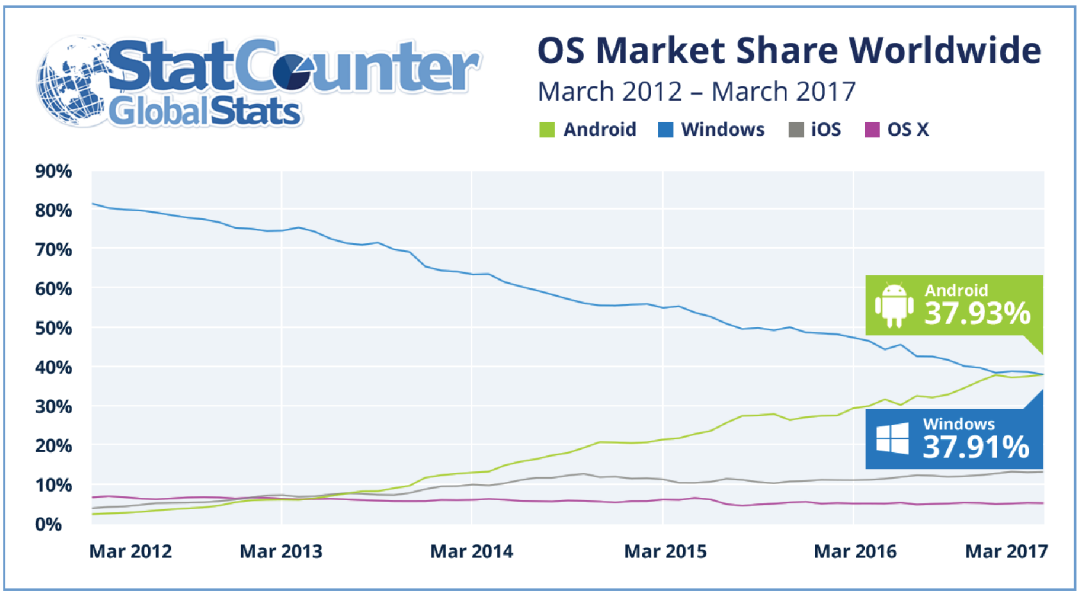


Figura 2 – Imagem da estatística de sistemas operacionais mais utilizados do mundo.

Fonte: Site http://gs.statcounter.com/press/android-overtakes-windows-for-first-time.

Uma parte da arquitetura do Android é dividida em 6 grandes componentes que podem ser visualizados na figura 3.

O kernel do Linux é utilizado como base para a plataforma e faz com que ela aproveite diversos recursos como, por exemplo, os principais recursos de segurança deste SO. A Hardware Abstraction Layer, é a camada responsável pela implementação de interfaces

para um tipo específico de componente de hardware, como áudio, bluetooth, câmera, sensores, etc fazendo uso de módulos de bibliotecas. A Android Runtime é otimizada para oferecer consumo mínimo de memória, fazendo a criação de diversas máquinas virtuais em dispositivo de baixa memória executando arquivos DEX, e a partir da API nível 21, é criada uma instância do Android Runtime para cada processo de aplicativo. Bibliotecas nativas que foram programadas em C e C++, são utilizadas em código nativo como vários serviços e componentes que fazem parte do sistema operacional Android, por isso se faz necessária a camada Native C/C++ Libraries. A camada Java API Framework é necessária pois os recursos do Android estão disponíveis por meio das APIs que foram desenvolvidas na linguagem Java. Por fim, a camada dos aplicativos do sistema, onde o Android já conta com diversos aplicativos importantes como aplicativo para gerenciamento de contas de email, navegação na internet e etc (DEVELOPER, 2018).

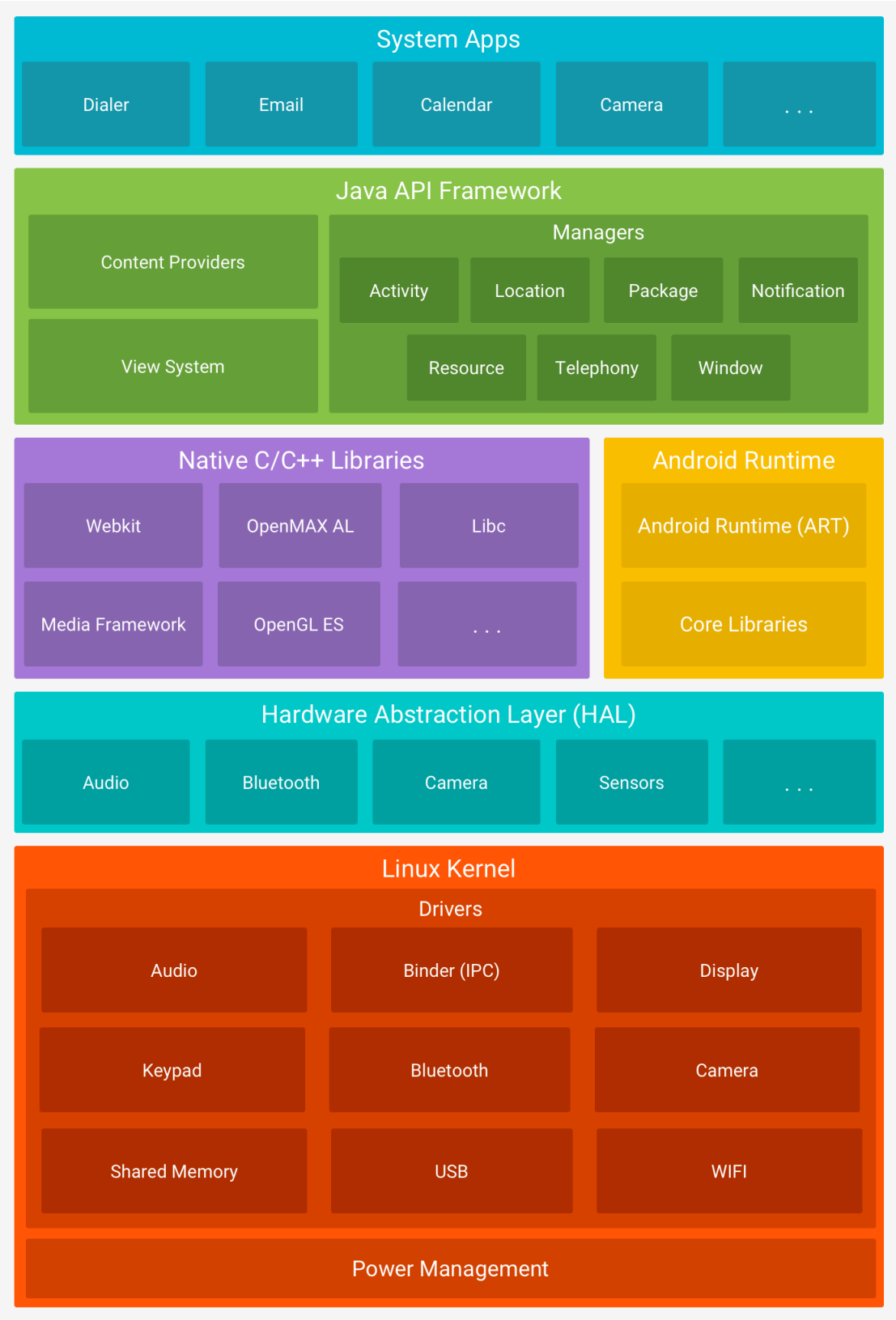


Figura 3 – A pilha de software do Android.

Fonte: Site https://developer.android.com/guide/platform/.

Capítulo 3

Materiais

3.1 Raspberry Pi

Do tamanho de um cartão de crédito e atualmente com um alto poder de processamento, o Raspberry Pi pode se conectar a um monitor, teclado e mouse e ser usado como um desktop. Além das diversas funcionalidades que ele nos permite desfrutar, tais como, navegação na internet, manipulação de planilhas e jogos, este mini computador é muito versátil para a utilização em projetos eletrônicos. Sua utilização é destinada a crianças e adultos do mundo inteiro e tem como finalidade trazer a aprendizagem em programação e criação de conteúdo digital (RASPBERRY, 2018).

Depois de muito desenvolvimento, somente em 2011, com demonstrações de Quake III e vídeos full HD/1080p, que a população pode ver a versão alpha da Raspberry Pi. Uma demonstração que a princípio demonstrou um alto poder de processamento em um computador de baixo custo (DENNIS, 2013).

Após um ano, o poderoso mini computador estava preparado para ser vendido ao público. Inicialmente sendo vendida em dois modelo, A e B. Com um preço de US $25 a placa do modelo A, não contava com uma porta Ethernet, porém, como consequência positiva, este modelo apresentava um consumo de energia muito mais considerável que o outro modelo. Já o segundo modelo, que contém a porta Ethernet foi sugerido o valor de US $35 (DENNIS, 2013).

Os modelos vendidos atualmente pela loja oficial online, disponível no site oficial[1], são: Raspberry Pi Zero, Raspberry Pi Zero W, Raspberry Pi Model A+, Raspberry Pi Model B+, Raspberry Pi 2 Model B, Raspberry Pi 3 Model B e Raspberry Pi 3 Model B+.

No desenvolvimento deste projeto foi utilizado o Raspberry Pi 3 Model B visível na figura 4, que substituiu o Raspberry Pi 2 Model B em fevereiro de 2016. Este modelo conta uma CPU Quad Core de 1.2GHz Broadcom BCM2837 de 64bit, trabalhando junto com 1GB de memória RAM, 4 portas USB 2.0, saída estéreo de 4 polos e porta de vídeo

[1] Produtos Raspberry Pi. Disponível em: <https://www.raspberrypi.org/products/>. Acessado em 31/07/2018.

composta, uma entrada para encaixe do Micro SD, entrada para fonte de energia Micro USB comutada atualizada até 2,5A e GPIO estendidos até 40 pinos, que podem ser melhor visualizados figura 5.

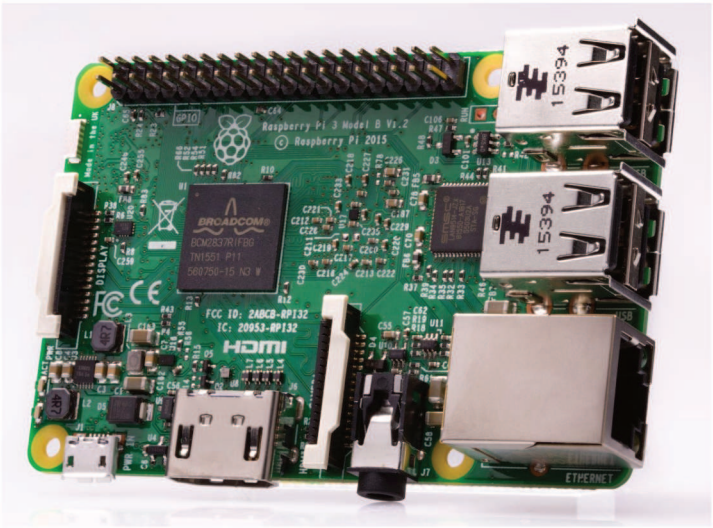


Figura 4 – Raspberry Pi 3 Model B.

Fonte: Site https://www.raspberrypi.org/products/raspberry-pi-3-model-b/#buy-now-modal.

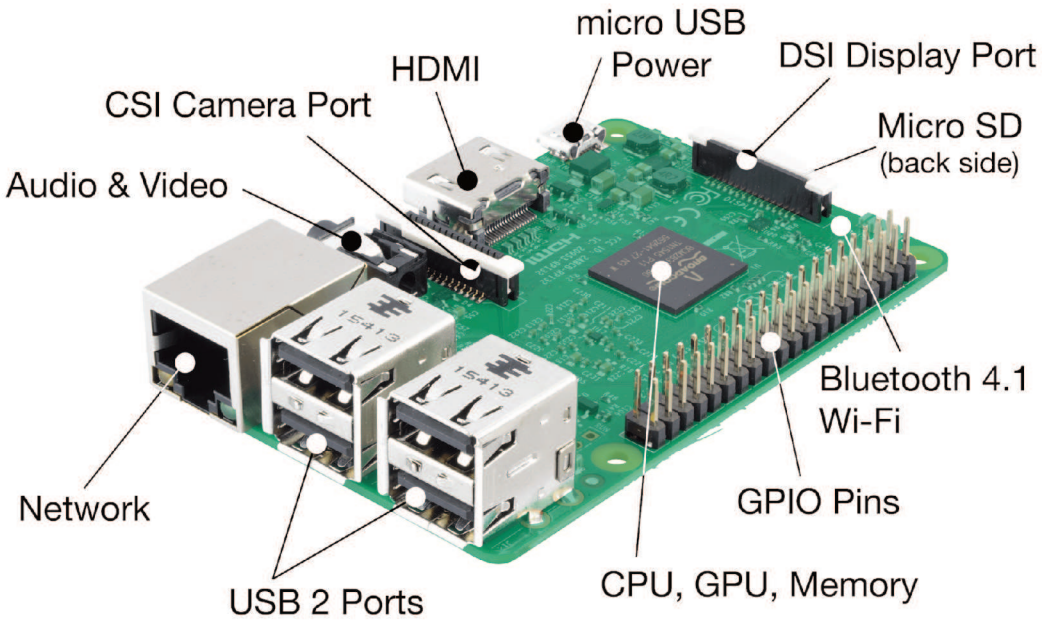


Figura 5 – Componentes da versão Pi 3 Model B.

Fonte: Site https://teekle.co.za/product/raspberry-pi-3-model-b/.

É possível conferir a numeração e o que cada GPIO da Raspberry Pi 3 Model B representa conforme figura 6:

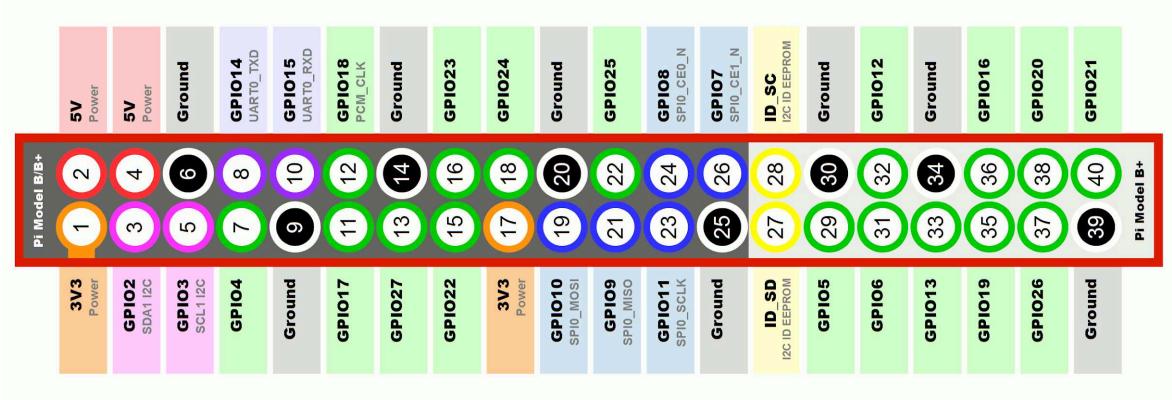


Figura 6 – Pinagem da versão Pi 3 Model B.

Fonte: Site https://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/.

Além de ser possível fazer a instalação de sistema operacional neste mini computador, há versões de sistemas que foram otimizadas para o funcionamento nesta CPU, como disponíveis no site oficial: Ubuntu Mate, Snappy Ubuntu Core, Windows 10 IOT Core, OSMC, Libreelec, Pinet, Risc OS, Wather Station, Ichigojam RPI.

Com essa gama de sistemas disponíveis, para este projeto foi utilizado o sistema Raspbian, por ser o sistema operacional oficial da Fundação e já vir com diversos pacotes pré-instalados para tornar mais prático e rápido a produção do produto. Dentre os diversos softwares instalados, vale citar softwares para educação, programação e uso geral, como Python, Scratch, Sonic Pi, Java, Mathematica e outros (RASPBIAN, 2018).

3.2 Push Button

Este dispositivo denominado push button, do inglês, botão de pressão, funcionando como uma chave, permitindo ou barrando a passagem da corrente. Este trabalha com o uso de botão para fazer o acionamento, abrindo ou fechando o circuito a qual está conectado.

Figura 7 – Push Button.

Diferentemente de um interruptor que muda de estado, permitindo o circuito estar aberto ou estar fechado até que seja acionado novamente, o push button mantém a troca do estado do circuito apenas durante o acionamento da peça, portanto, ele mantém a ação de contato apenas temporariamente.

Há alguns tipos de push buttons, mas os mais comuns são os que permanecem em seus estados padrão abertos ou fechados, ou seja, no caso do push button com estado padrão aberto, quando é acionado muda de estado fechando o circuito, e o outro push button trabalha ao contrário, por padrão mantém o circuito já fechado e com seu acionamento abre o circuito.

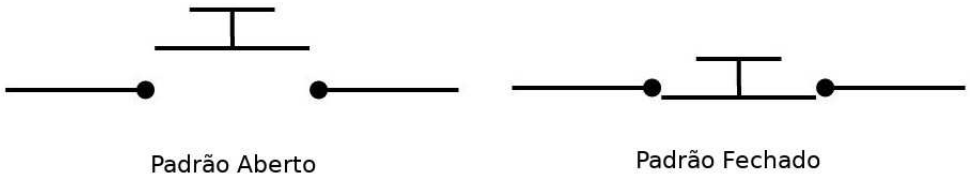


Figura 8 – Estado padrão push buttons.

Como é possível perceber, o primeiro circuito da figura 8 é do push button com estado padrão aberto, pois quando acionado permite a passagem da corrente, assim como o segundo circuito é sempre permitido a passagem da corrente e quando pressionado o circuito fica aberto impedindo a fluidez da corrente.

Neste projeto foi utilizado um push button igual o da figura 7 que tem como padrão o estado aberto e de acordo com a figura 9, fecha o circuito montado entre todos os terminais quando pressionado.

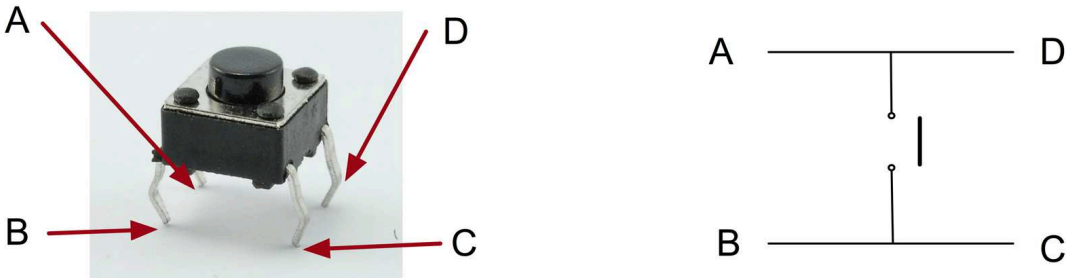


Figura 9 – Push Button com quatro terminais.

Fonte: Site http://razzpisampler.oreilly.com/ch07.html.

3.3 Headset

Headset é uma palavra originária do inglês que significa fone de ouvido e é muito utilizada para designar fones de ouvido que possuem um controlador de volume além de um microfone acoplado. Também apelidado de fone operador, fone de cabeça, headfone ou ainda handsfree é um produto muito utilizado no mundo de telemarketing que vai plugado a um microcomputador buscando trazer mais agilidade na hora do operador exercer sua profissão. Sua difusão vem se dando pelo fato de poder exercer diversas tarefas com ele, sem estar com as mãos ocupadas pelo headset e vem ganhando cada vez mais destaque com fones mais sofisticado sendo desenvolvidos também para o uso pessoal (CANALTECH, 2018).

Para fazer o protótipo deste produto foi utilizado um headset da Multilaser que utiliza para transferência de dados uma porta usb. A utilização deste headset simula tanto o auto falante quanto o microfone do interfone, consequentemente foram feitas configurações específicas no sistema operacional da Raspberry para configuração deste fone como principal. Cabe salientar que outro fone poderia ser utilizado, sendo também possível fazer o uso de alto falantes e microfones separados. O headset utilizado se assemelha com o da figura 10.

Figura 10 – Headset.

3.4 Smartphone e Android Studio

Para o desenvolvimento do aplicativo foi utilizado a IDE Android Studio que depois de muito trabalho por parte de seus desenvolvedores se encontra na versão 3.1.3. O desenvol-

vimento da IDE é exercido pela equipe de desenvolvedores do Google e é disponibilizada de forma gratuita para os três sistemas operacionais para desktop mais comuns: Windows, Mac e Linux.

Figura 11 – Tela inicial da IDE Android Studio.

No sistema foi utilizado a linguagem de programação Java para o desenvolvimento do aplicativo, juntamente com o uso da linguagem de marcação XML para construção do layout das telas. Atualmente é possível fazer uso também da linguagem Kotlin no lugar do Java ou usá-la junto ao Java.

A IDE conta com diversas ferramentas que buscam aumentar a produtividade durante um trabalho como um analisador do aplicativo, editor inteligente, um sistema de construção flexível e outras ferramentas disponíveis.

É possível fazer diversas alterações na interface do software de desenvolvimento buscando deixar o ambiente de trabalho mais cômodo possível para seus usuários, mas em geral, a interface principal pode ser dividida conforme mostra a figura 12, sendo dividida em 6 partes:

- Barra de ferramentas: onde fica localizada algumas ferramentas que a IDE disponibiliza;
- Área de edição: local utilizado para inserção do código que gerará o aplicativo;

- ❏ Preview: na edição do layout por meio de código, no preview é possível ver as alterações gráficas em tempo real;
- ❏ Área de navegação: local que permite a visualização e manipulação dos arquivos que fazem parte do projeto;
- ❏ Caminho do arquivo: estrutura de pastas do caminho do arquivo que está sendo editado no momento;
- ❏ Área de depuração: utilização para depuração do aplicativo.

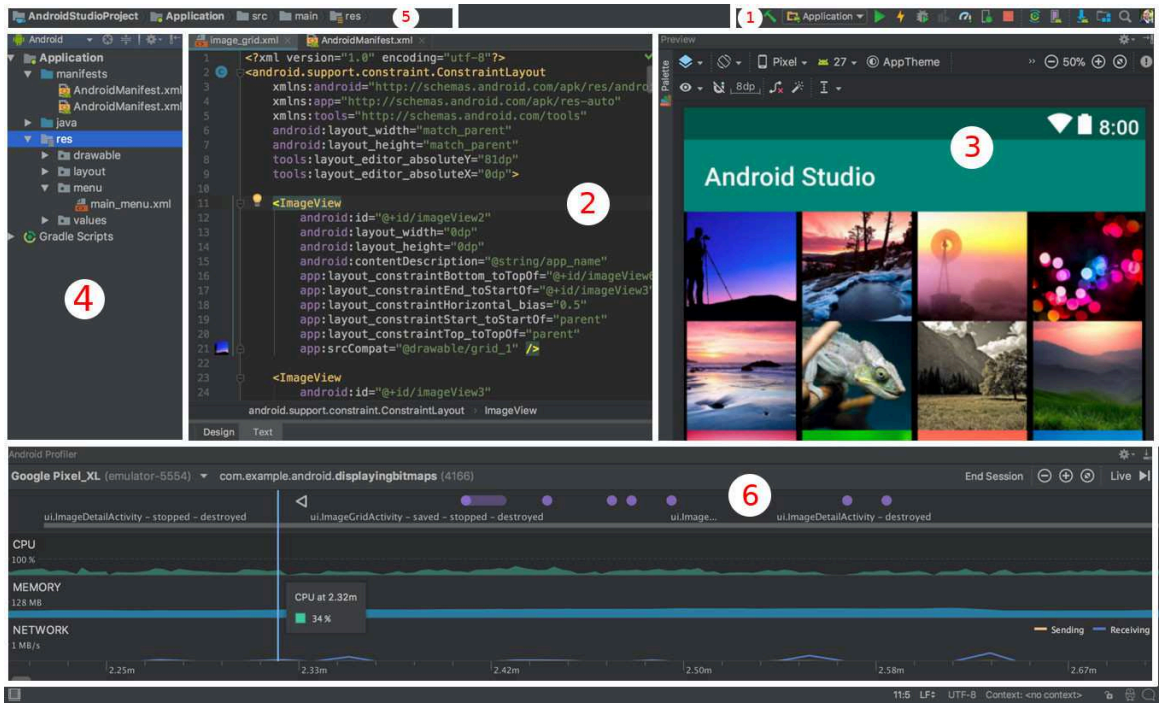


Figura 12 – Interface do Android Studio.

Fonte: Site https://developer.android.com/studio/.

Outra funcionalidade excepcional que é disponibilizada pela IDE é que ela acompanha um emulador de um aparelho Android para testes no aplicativo desenvolvido, nos permitindo escolher desde qual versão Android queremos emular, até dimensões de tela e também emulação de uma TV Android.

O aplicativo deste projeto denominado Interfone Virtual foi desenvolvido para funcionar com um Sdk mínimo na versão 17, que é o Android 4.2.2 Jelly Bean e com um Sdk target sendo o de versão 27, Android 8.1 Oreo. O aplicativo foi testado em algumas versões do Android e mais especificamente em um aparelho físico que acompanhava o Android 7.1.2 Nougat de uma ROM com Android puro.

Capítulo 4

Implementação e desenvolvimento

4.1 Firebase

4.1.1 Configuração do sistema Firebase

Para fazer a utilização do serviço de banco de dados disponibilizado pelo Firebase é necessário fazer o vínculo da aplicação desenvolvida com a plataforma do Google. A conta a ser utilizada para criação do projeto no Firebase é a própria conta Google que é utilizado em diversos sites. Então com a conta Google pronta é necessário entrar no site[1] para fazer a criação do projeto Firebase e, para criar um projeto é necessário apenas um nome para o projeto, a região de faturamento e do Analytics (outro serviço oferecido pela plataforma) e aceitar os termos conforme figura 13.

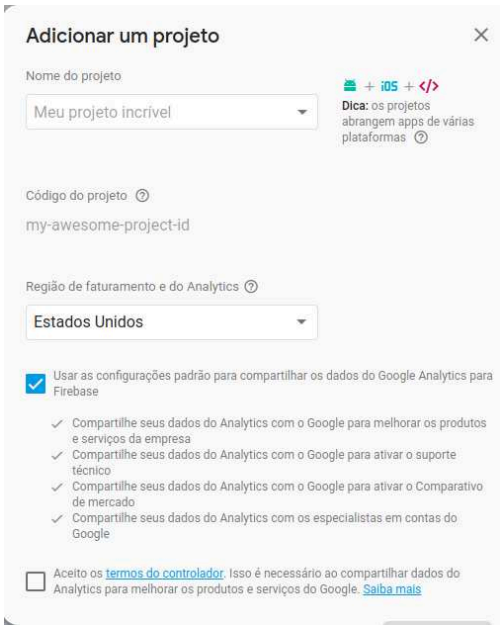


Figura 13 – Tela de criação do projeto Firebase.

[1] Firebase Console. Disponível em: <https://console.firebase.google.com>. Acessado em 02/08/2018.

A plataforma tem suporte nativo para aplicações web, IOS e Android. Porém como foi utilizado a linguagem Python no desenvolvimento do sistema na Raspberry, foi utilizado uma biblioteca open source para fazer a conexão com o Firebase.

A plataforma conta com a sessão Authentication que faz o controle de acesso aos serviços disponíveis. Há diversos modos de autenticação como smartphone, Google, Play Games, Facebook, Twitter, GitHub, Anônimo e usuário/senha. Para este projeto foi utilizado a forma com usuário e senha para garantir uma parte da segurança ao utilizar o sistema. Então foi configurado neste módulo a ativação da autenticação por email e senha e criado um usuário para o protótipo.

Passando para a sessão de Database e ao solicitar para criação do banco de dados em tempo real é pedido a regra de segurança que será aplicada como na figura 14. É escolhido o modo bloqueado que impede todos os tipos de acesso ao banco, porém será alterado logo em seguida.

Figura 14 – Tela de escolha de regras de segurança.

Com o banco criado é então alterado a regra de segurança para permitir acesso de aplicações que fazem autenticação, para isto é necessário ir no item “Regras” e alterar o código existente pelo mostrado na figura 15. Fazendo então a publicação das novas regras, o banco está pronto para ser usado.

Para gerar a conexão entre o aplicativo Android e o sistema do Firebase é necessário adicionar o Firebase ao app. Para isto no menu "Project Overview"há um seguimento de passos necessários, onde existe um guia criado pela própria plataforma instruindo o que é necessário fazer. Dentre os passos há uma parte onde é preciso gerar um certificado de assinatura de depuração SHA-1 na IDE Android Studio e outro passo onde é feito

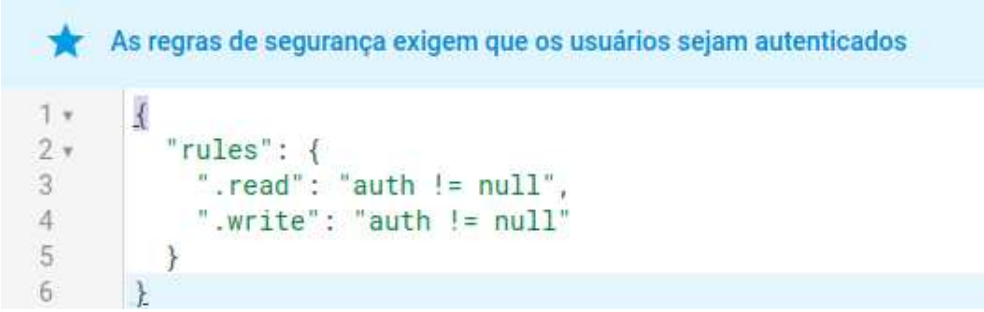


Figura 15 – Código de autenticação para acessar o banco.

o download de arquivos do Firebase que devem estar presentes dentro dos arquivos do aplicativo.

Já para acesso via Python pela Raspberry Pi, são utilizados os dados como se fosse pela web, utilizando o domínio de autenticação, URL do banco e a chave da API Web, encontrada nas configurações do projeto. Demais informações sobre a API utilizada para acesso via Python serão detalhados no tópico 4.2.4.

4.1.2 Modelo do banco de dados

Como o banco de dados Realtime disponibilizado pelo Firebase é um banco não relacional e adota um formato de árvore, onde cada registro final, conhecido também como um registro folha, é definido por uma chave e um conteúdo. É possível ter uma melhor visualização do formato do banco na figura 16.

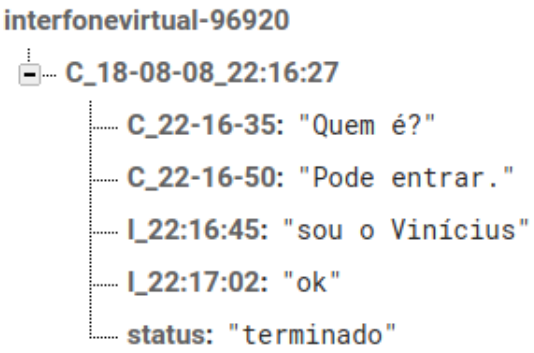


Figura 16 – Registro salvo no banco Firebase.

Pelo fato do banco adotar esta estrutura, foi desenhado um modelo para poder obter um melhor aproveitamento do banco. Neste modelo foi definido o formato que cada registro será gravado no banco.

Para um registro que identifique uma chamada, será criado um nó com o seguinte padrão: C_ano-mes-dia_hora:minuto:segundo. Este padrão foi escolhido para facilitar

na hora de visualizar os dados no banco pois, o C inicial é de chamada e os dados subsequentes são necessários para controle de quando alguém foi a residência. A criação do nó chamada, sempre vem acompanhado de uma nó folha que é necessário para fazer o controle de estados da chamada, tendo como chave deste nó o nome status.

Os subnós referentes às mensagens trocadas seguem o seguinte formato: I_hora:minuto:segundo. Este padrão é alterado para quando um nó é registrado pelo celular, tendo como inicial a letra C e a troca dos dois pontos por hífen.

Com este modelo é possível saber a ordem das mensagens trocadas, assim como o dia, mês e ano do registro, possibilitando por exemplo, ter um controle de quando uma pessoa foi a residência.

4.2 Raspberry Pi 3

4.2.1 Configuração do push button com a Raspberry Pi

Para simulação do botão que é acionado no interfone, foi utilizado um push button. O esquemático da montagem se deu como na figura 17, que foi construída utilizando o software Fritzing disponível para download em seu site[2], um open-source que além de outras funções, auxilia na construção de esquemático de projetos eletrônicos.

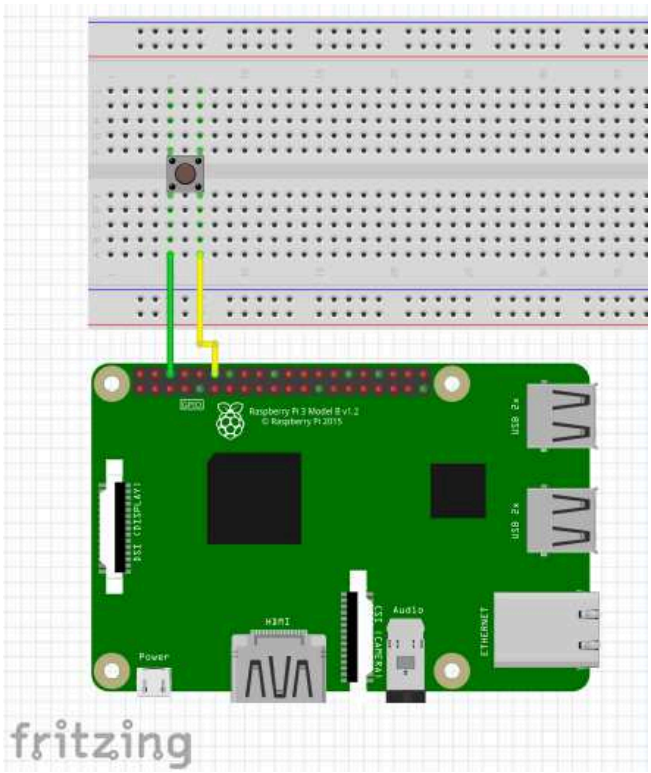


Figura 17 – Conexão do push button.

[2] Fritzing. Disponível em: <http://fritzing.org/home/>. Acessado em 02/08/2018.

Analisando a pinagem da Raspberry Pi 3 Model B, é possível ver que de acordo com a figura 9, tópico 3.2, os terminais do push button fecha curto entre os pinos GPIO 18 e ground da raspberry quando o botão for pressionado. O pino de entrada GPIO 18 é normalmente configurado para 3.3V pela configuração padrão da Raspberry Pi, portanto, quando o botão for pressionado o GPIO passará a ter a tensão de ground. Concluindo, ao se ler o valor da entrada do GPIO, caso for retornado false, significará que o botão estará sendo pressionado (O'REILLY, 2013).

4.2.2 Configuração dos pacotes e versões de softwares na Raspberry Pi

Buscando uma maior compatibilidade com as APIs e softwares existes, o primeiro passo executado no microprocessador foi a atualização de diversos pacotes assim como download de versões mais recentes da linguagem Python, por exemplo.

Como o Raspbian é um sistema operacional baseado no Debian, foi começado a atualização pelo comando a seguir:

```
pi@raspberrypi:~ $ sudo apt-get update && sudo apt-get upgrade -y
```

Este comando é utilizado para fazer a atualização dos pacotes existentes no sistema com a ultima versão disponivel em seus repositórios. A primeira parte do comando faz uso do apt que é o gerenciador de pacotes padrão do Debian junto com a opção update que faz a sincronização das versões de pacotes do sistema com os mais atuais disponíveis. A segunda parte com a utilização do upgrade é responsável então por fazer o download e instalação dos pacotes que têm versões mais novas disponíveis e utilizando o -y para aceitação do download dos pacotes auxiliares.

Para execução do sistema desenvolvido, na Raspberry Pi se utiliza o Python. Devido à biblioteca Pyrebase utilizada no sistema ter sido escrita para o Python 3 e, como informado em sua documentação não irá funcionar corretamente com o Python 2, foi definido utilizar o Python 3 em uma versão mais atual. Devido a esta circunstância foi escolhido o Python 3 versão 3.6.5 e feita sua instalação. Como está versão não está disponível nos repositórios padrões, foi utilizado a versão disponibilizada pelo site[3] oficial da linguagem.

Antes da instalação da versão mais recente do Python, foi instalado ferramentas necessárias utilizando o gerenciador de pacotes apt:

```
pi@raspberrypi:~ $ sudo apt-get install build-essential tk-dev libncurses5-dev libncursesw5-dev libreadline6-dev libdb5.3-dev libgdbm-dev libsqlite3-dev libssl-dev libbz2-dev libexpat1-dev liblzma-dev zlib1g-dev
```

[3] Python. Disponível em: <https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tar.xz>. Acessado em 06/08/2018.

E então feito o download do arquivo compactado utilizando o wget e efetuada a instalação conforme instruído pelo arquivo "README.rst" presente dentro do pacote baixado.

```
pi@raspberrypi:~ $ wget https://www.python.org/ftp/python/3.6.5/Python- 3.6.5.tar.xz
pi@raspberrypi:~ $ tar xf Python-3.6.5.tar.xz
pi@raspberrypi:~ $ cd Python-3.6.5
pi@raspberrypi:~/Python-3.6.5 $ ./configure
pi@raspberrypi:~/Python-3.6.5 $ make
pi@raspberrypi:~/Python-3.6.5 $ sudo make install
```

Por fim, antes de começar a instalação das bibliotecas utilizados no projeto, foi atualizado por último o gerenciador de pacotes do Python 3, o pip3:

```
pi@raspberrypi:~ $ sudo pip3 install –upgrade pip
```

É necessário a instalação de algumas bibliotecas para fazer a utilização de alguns recursos necessários, como acesso ao Firebase, já que a empresa não oferece suporte a linguagem Python, foi utilizada uma biblioteca open-source de terceiro, utilização dos GPIOs disponíveis na Raspberry Pi entre outros módulos.

O primeiro pacote é o pyaudio que fornece ligações em Python para o PortAudio, que é uma biblioteca que faz o gerenciamento de entrada e saída de áudio da plataforma. Como a versão do Python foi então atualizada para versão 3.6.5 foi decidido utilizar o código fonte direto do site[4] oficial. Com a utilização da ferramenta git, foi clonado do repositório o código, feito a instalação de bibliotecas auxiliares e então instalado conforme comandos a seguir:

```
pi@raspberrypi:~ $ sudo git clone http://people.csail.mit.edu/hubert/git/ pyaudio.git
pi@raspberrypi:~ $ sudo apt-get install libportaudio0 libportaudio2 libportaudiocpp0
portaudio19-dev
pi@raspberrypi:~ $ cd pyaudio/
pi@raspberrypi:~/pyaudio $ sudo python3 setup.py install
```

Agora utilizando o gerenciador de pacotes do Python 3, o pip3, é instalado três bibliotecas necessárias: SpeechRecognition necessária para fazer o reconhecimento de voz utilizado no momento da fala do visitante, disponível no site[5] oficial, RPi.GPIO utilizada para fazer a leitura do pino onde foi ligado o push button, disponível no site[6] oficial e por fim, a biblioteca responsável pelo acesso ao Firebase pela linguagem Python, o Pyrebase, disponível no repositório GitHub[7].

```
pi@raspberrypi:~ $ sudo pip3 install pyrebase
```

[4] PyAudio. Disponível em: <https://people.csail.mit.edu/hubert/pyaudio/>. Acessado em 06/08/2018.
[5] SpeechRecognition. Disponível em: <https://pypi.org/project/SpeechRecognition/>. Acessado em 06/08/2018.
[6] RPi.GPIO. Disponível em: <https://pypi.org/project/RPi.GPIO/>. Acessado em 06/08/2018.
[7] Pyrebase. Disponível em: <https://github.com/thisbejim/Pyrebase>. Acessado em 06/08/2018.

```
pi@raspberrypi:~ $ sudo pip3 install SpeechRecognition
pi@raspberrypi:~ $ sudo pip3 install RPi.GPIO
```

A biblioteca do Pyrebase emite um erro em tempo de execução quando é tentado fazer a fechamento de uma escutado do banco. Porém, como este erro não influencia no processamento do sistema, é possível apenas ignorá-lo. Para isto será feito a alteração da classe e método do Pyrebase que gera este erro, alterando o método close(self), da classe ClosableSSEClient(SSEClient), que se encontram no arquivo /usr/local/lib/python3.6/site-packages/pyrebase/pyrebase.py. Este método deve ser reescrito da seguinte forma:

```
def close(self):
    self.should_connect = False
    self.retry = 0
    try:
        self.resp.raw._fp.fp.raw._sock.shutdown(socket.SHUT_RDWR)
        self.resp.raw._fp.fp.raw._sock.close()
    except AttributeError:
        pass
```

No método original apenas foi adicionado a cláusula try para caso aconteça o erro específico, permita que o sistema continue funcionando normalmente.

Para reprodução dos áudios de resposta, foi utilizado um software próprio para isto, fazendo com que o programa em Python reproduza a resposta do morador utilizando este software. A instalação deste e de um programa auxiliar foi concretizada pelos seguinte comandos:

```
pi@raspberrypi:~ $ sudo apt-get install mpg321
pi@raspberrypi:~ $ sudo apt-get install flac
```

Finalizando assim a instalação dos softwares e bibliotecas auxiliares que foram utilizadas em todo o projeto.

4.2.3 Arquivos de áudio de respostas

Como o sistema que roda na Raspberry Pi reproduz respostas pré-definidas, quando um novo registro é salvo no banco com a resposta do usuário morador o sistema reproduz o áudio correspondente ao registro salvo. Foram definidas 8 respostas:

- ❏ Quem é?
- ❏ Pode entrar.
- ❏ Por favor, aguarde um momento.
- ❏ Estou indo!

- ❏ Precisa de documento?
- ❏ Com quem deseja falar?
- ❏ Não se encontra no momento.
- ❏ Se possível, volte mais tarde.

Os áudios com estas devidas respostas foram gravados e devem estar presentes no diretório: /home/pi/audiosTCC/.

```
pi@raspberrypi:~ $ ls /home/pi/audiosTCC/
com_quem_deseja_falar.mp3          por_favor_aguarde_um_momento.mp3
estou_indo.mp3                     precisa_de_documento.mp3
nao_se_encontra_no_momento.mp3     quem_eh.mp3
pode_entrar.mp3                    se_possivel_volte_mais_tarde.mp3
```

4.2.4 Configuração do headset na Raspberry Pi

Como a Raspberry Pi já vem com uma saída de áudio definida como padrão e ao se plugar o fone neste caso na entrada usb, não é alterada a saída automaticamente, portanto será necessário fazer a alteração da saída de áudio padrão.

Primeiro passo é verificar se o sistema operacional reconhece o fone de ouvido inserido. Utilizando o comando lsusb que entrega informações sobre as portas e firmwares dos dispositivos conectados, será possível conferir se foi reconhecido.

```
pi@raspberrypi:~ $ lsusb
Bus 001 Device 004: ID 0d8c:013c C-Media Electronics, Inc. CM108 Audio Controller
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp. SMSC9512/9514
Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp. SMC9514 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

É possível ver que foi reconhecido como o primeiro dispositivo listado o “Bus 001 Device 004”. Tendo reconhecido é necessário alterar o arquivo de configuração de audio do sistema, o alsa.conf localizado em /usr/share/alsa/. Nele será necessário alterar os valores das propriedades defaults.ctl.card, defaults.pcm.card, defaults.pcm.device e defaults.pcm.subdevice para 1, 1, 0 e 0 respectivamente.

Com isso configurado é possível ver se a entrada foi mudada utilizando o comando alsamixer que é usado para configurar som e ajustar o volume.

4.2.5 Software de interação

Tendo as configurações até aqui efetuadas corretamente, com as devidas bibliotecas instaladas e softwares atualizados, é necessário fazer a iniciação do programa desenvolvido para o mini computador.

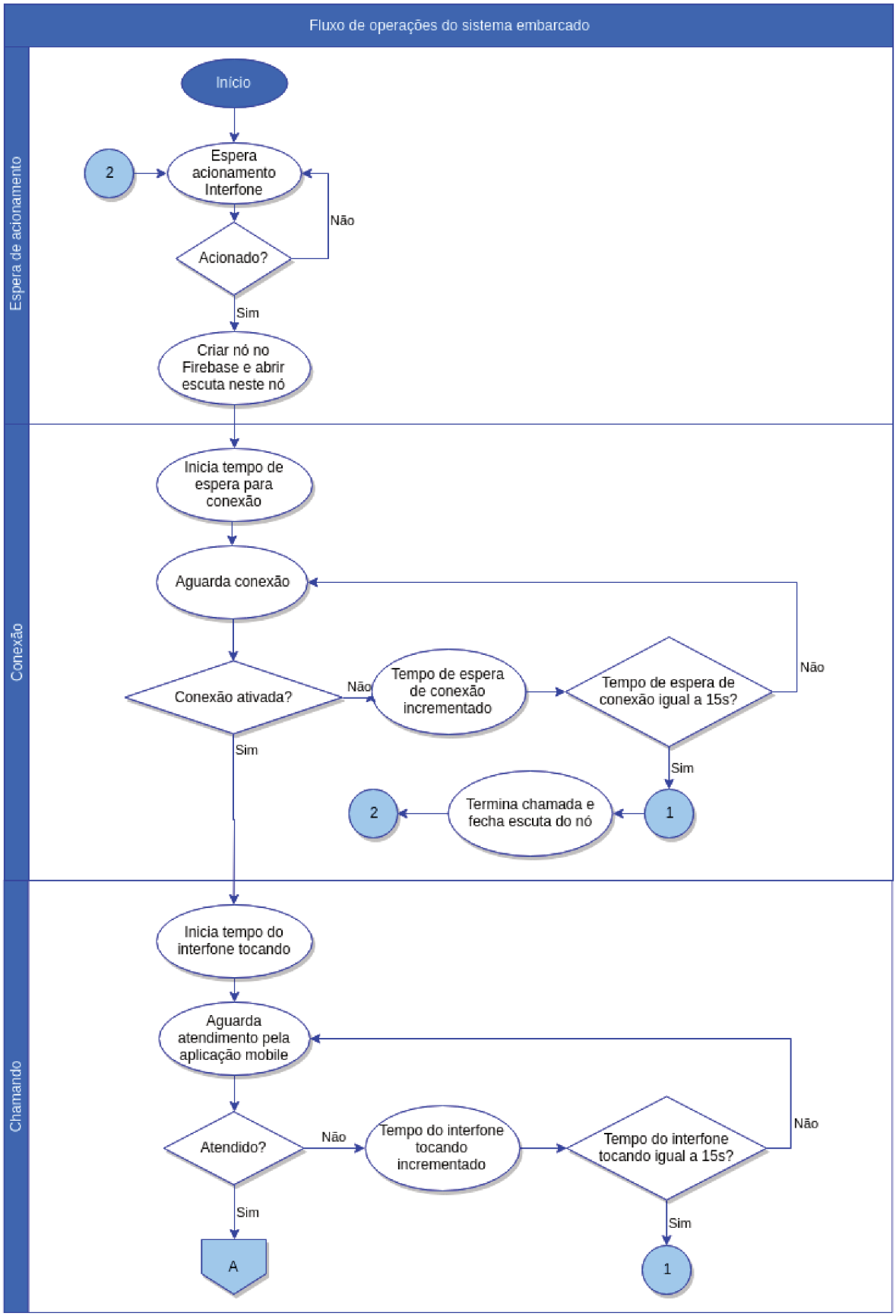
O programa desenvolvido para ficar em execução na Raspberry Pi se chama "main.py" e foi desenvolvido na linguagem Python. Está linguagem foi escolhida devido ao grande número de bibliotecas suporte que tem disponibilizadas para uso, além de ser uma linguagem com sintaxe clara e direta, sendo uma das principais linguagens de programação utilizada pelo Raspbian.

É possível ter uma melhor visão do fluxo de operações que são executadas pelo sistema, visualizando o fluxograma conforme apresentado pela figura 18.

Como é possível ver pelo fluxograma, o sistema é dividido em quatro estados que são sincronizados com a aplicação mobile por meio do banco Firebase, porém antes de iniciar estes passos é necessário fazer a iniciação dos objetos que foram utilizados e fazer o fornecimento das chaves de segurança que são necessárias para que o software consiga fazer o acesso ao Firebase. Estas chaves estão disponíveis nas configurações do projeto criado no Firebase, e na parte do menu destinada ao banco de dados será possível encontrar o link utilizado para acesso ao banco via web. Além da obtenção destas chaves, ainda na iniciação foi feito as configurações necessárias para que os pinos de GPIO façam a leitura do push button.

```
config = {
"apiKey": "apiKey",
"authDomain": "projectId.firebaseapp.com",
"databaseURL": "https://databaseName.firebaseio.com/",
"storageBucket": "projectId.appspot.com",
}
firebase = pyrebase.initialize_app(config)
auth = firebase.auth()
user = auth.sign_in_with_email_and_password('blabla@gmail.com', '1234')
db = firebase.database()
```

A partir disso o software passa a permanecer no fluxo apresentado no diagrama. Na primeira fase, na espera do acionamento a Raspberry fica num ciclo infinito lendo o valor do GPIO definido para o push button. Caso a leitura se mantenha a mesma, neste sistema em 3.3V por causa do padrão da Raspberry Pi, o sistema simplesmente faz novamente a leitura até o momento em que o push button é apertado e então o pino do push button entra em contato com o ground e aciona a condição definida no programa. Com a condição aceita o software para de ler o GPIO, cria um nó para esta ocorrência no Firebase e passa para fase de espera de conexão.


Fluxo de operações do sistema embarcado
Espera de acionamento
Início
2
Espera acionamento Interfone
Não
Acionado?
Sim
Criar nó no Firebase e abrir escuta neste nó
Conexão
Inicia tempo de espera para conexão
Aguarda conexão
Conexão ativada?
Não
Tempo de espera de conexão incrementado
Tempo de espera de conexão igual a 15s?
Não
Sim
Sim
2
Termina chamada e fecha escuta do nó
1
Chamando
Inicia tempo do interfone tocando
Aguarda atendimento pela aplicação mobile
Atendido?
Não
Tempo do interfone tocando incrementado
Tempo do interfone tocando igual a 15s?
Não
Sim
Sim
A
1

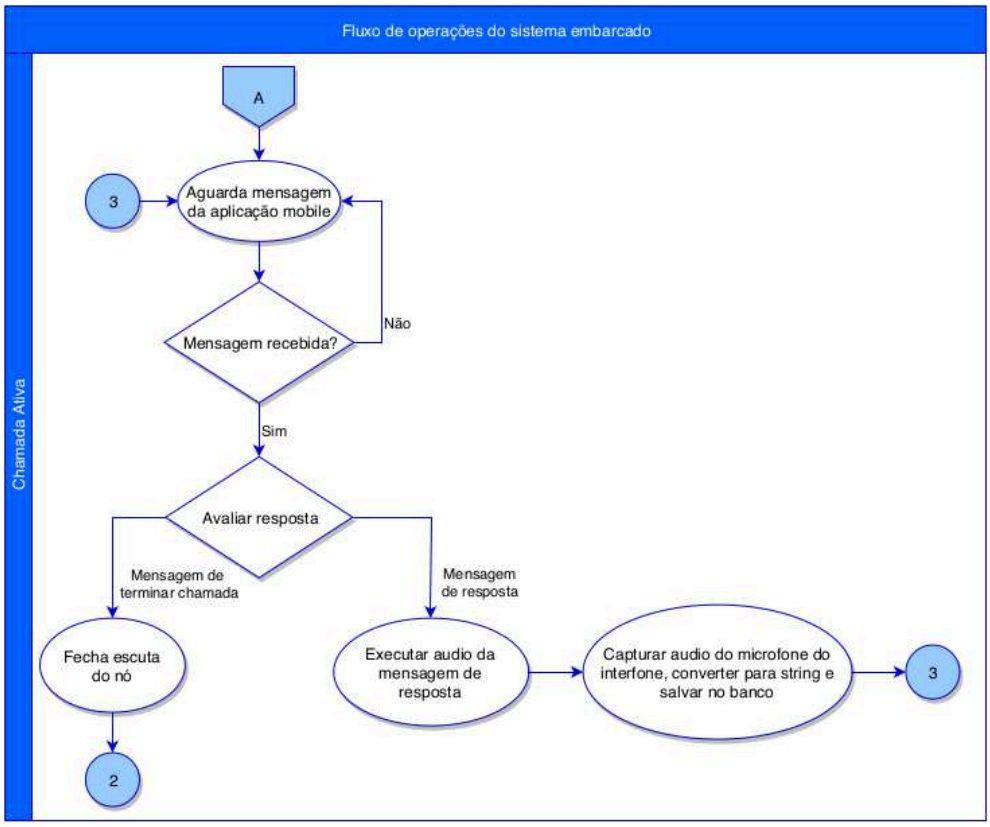


Figura 18 – Fluxo de operações do sistema embarcado.

A fase de espera de conexão se faz necessária pelo fato de nem sempre o aplicativo mobile estar conectado a internet e foi pensada mais no fato de fazer a sincronização entre o software embarcado e o aplicativo. Ao criar o nó no banco, o próprio Firebase aciona o aplicativo, mas esse acionamento pode demorar alguns segundos e caso se iniciasse direto a fase de chamada, por causa deste atraso de acionamento do aplicativo, o interfone iria terminar o tempo de "chamando" antes do aplicativo terminar o seu tempo de "chamando". Pensando nisso foi criado esta fase de espera de conexão, que aguarda por 15 segundo para que o aplicativo seja notificado e percebendo o novo registro do banco, o aplicativo muda o status da chamada para chamando e começa a "chamar" no smartphone, e com a mudança do status da chamada no banco, a Raspberry Pi passa então para o estado de espera de atendimento, o chamando.

A terceira fase do sistema é executada enquanto o aplicativo móvel está "chamando". Ela se assemelha com a fase anterior. A Raspberry Pi fica aguardando a mudança de status do registro no banco, que é alterada pelo aplicativo mobile quando o usuário atende a chamada, e então passa para a fase da chamada ativa, ou caso se passe os 15 segundos definidos para ser o tempo que o interfone ficará chamando e não tenha sido feito o atendimento, o software embarcado muda o status do registro no banco para terminado e volta para o estado inicial esperando pelo aperto no botão. Isso também ocorre na fase de aguardar conexão, caso o aplicativo não seja notificado sobre o novo registro dentro

dos 15 segundo do tempo de espera pela conexão.

Por fim o sistema entra na fase da chamada ativa, onde inicialmente fica aguardando a primeira mensagem ser enviada pelo aplicativo mobile, pois quando o morador atende o interfone, normalmente é o primeiro a iniciar a comunicação com o visitante, que de costume é perguntar qual o nome da pessoa que acionou o interfone de sua residência. Ao identificar a mensagem salva no banco, sendo a iniciação da conversa pelo morador, o software embarcado avalia está resposta, pois o morador pode ao invés de mandar uma mensagem de comunicação, enviar um comando para desligar a chamada, e neste caso o software embarcado simplesmente volta para o estado inicial de espera do aperto do push button. Mas, caso seja realmente uma resposta de comunicação, o programa busca o áudio correspondente a mensagem enviada pelo morador e executa este áudio para o visitante, onde, após o término do áudio, é de costume o visitante enviar sua resposta, portanto a Raspberry Pi passa a capturar o áudio do microfone e convertê-lo para texto por meio da biblioteca utilizada. Feito isto, o sistema salva esta resposta ao banco, que será apresentada ao morador por meio do aplicativo e volta a fase de aguardar a mensagem do morador.

Para execução deste software é necessário fazer sua inicialização utilizando a linguagem Python então atualizada. O código a seguir mostra o comando necessário para isto:

```
pi@raspberrypi:~ $ sudo python3 main.py

Aguardando pressionar o botao...
Leitura do sensor
Aguardando pressionar o botao...
Leitura do sensor
```

4.3 Desenvolvimento do aplicativo interfone virtual

4.3.1 Estruturação do aplicativo

Como um dos objetivos deste projeto é trazer um nível a mais de conforto para cada morador residencial, e como atualmente o smartphone faz quase de tudo e está geralmente perto de seu proprietário, o desenvolver deste sistema passou para a fase final, construção do aplicativo para smartphones que tem como seu sistema operacional o Android.

A criação deste aplicativo foi feita utilizando a IDE Android Studio, juntamente com a linguagem de programação Java para o desenvolvimento da lógica de programação e a utilização da linguagem de marcação XML para montagem do visual das telas que é por onde o usuário interage com o aplicativo.

Em resumo, o principalmente componente de criação de uma aplicação Android é uma Activity, pois, todo aplicativo desenvolvido para o sistema Android, começa por

uma Activity. Ela pode ser definida basicamente como uma classe controladora de uma UI (User Interface). No Android Studio, ao fazer a criação de uma nova Activity, será gerado uma classe Java e um arquivo de layout XML. E, por essa classe Java que será possível interagir com os elementos construídos na interface XML.

O desenvolvimento desta aplicação mobile, que recebeu o nome de Interfone Virtual, fez o uso de três Activitys sendo elas: a principal que é apresentada como primeira tela do aplicativo, uma que surge no momento em que o interfone está sendo tocado e outra para fazer a conversação com o visitante. Além destas três principais Activitys, foi desenvolvido também de suma importância, uma classe Java que cuida do trabalho em segundo plano do aplicativo, esperando pelo acionamento via Firebase.

Pela figura 19 é possível ver como foi feita a organização dos arquivos no aplicativo desenvolvido. Percebendo que para cada uma das três Activitys na pasta layout, há uma classe Java, dentro da pasta activity que faz referência a elas.

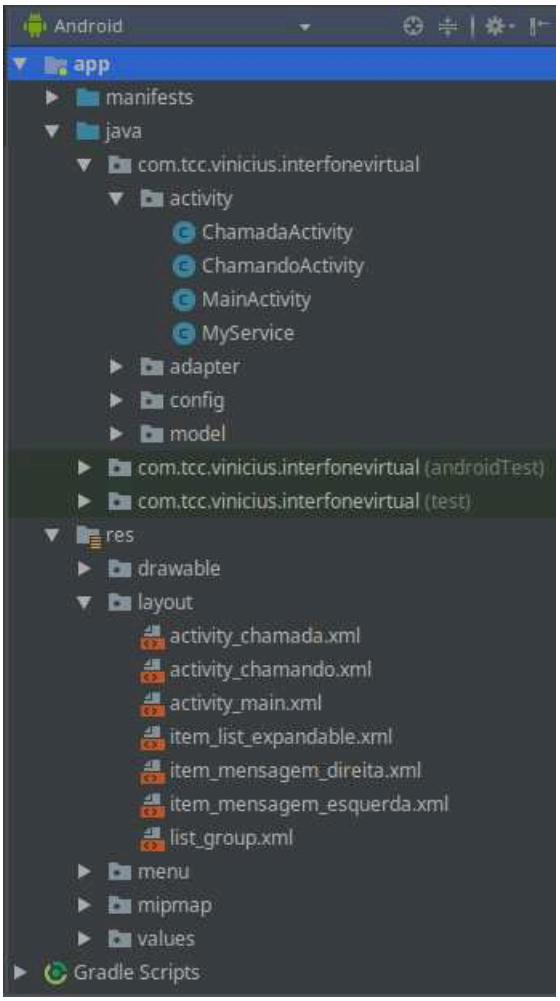
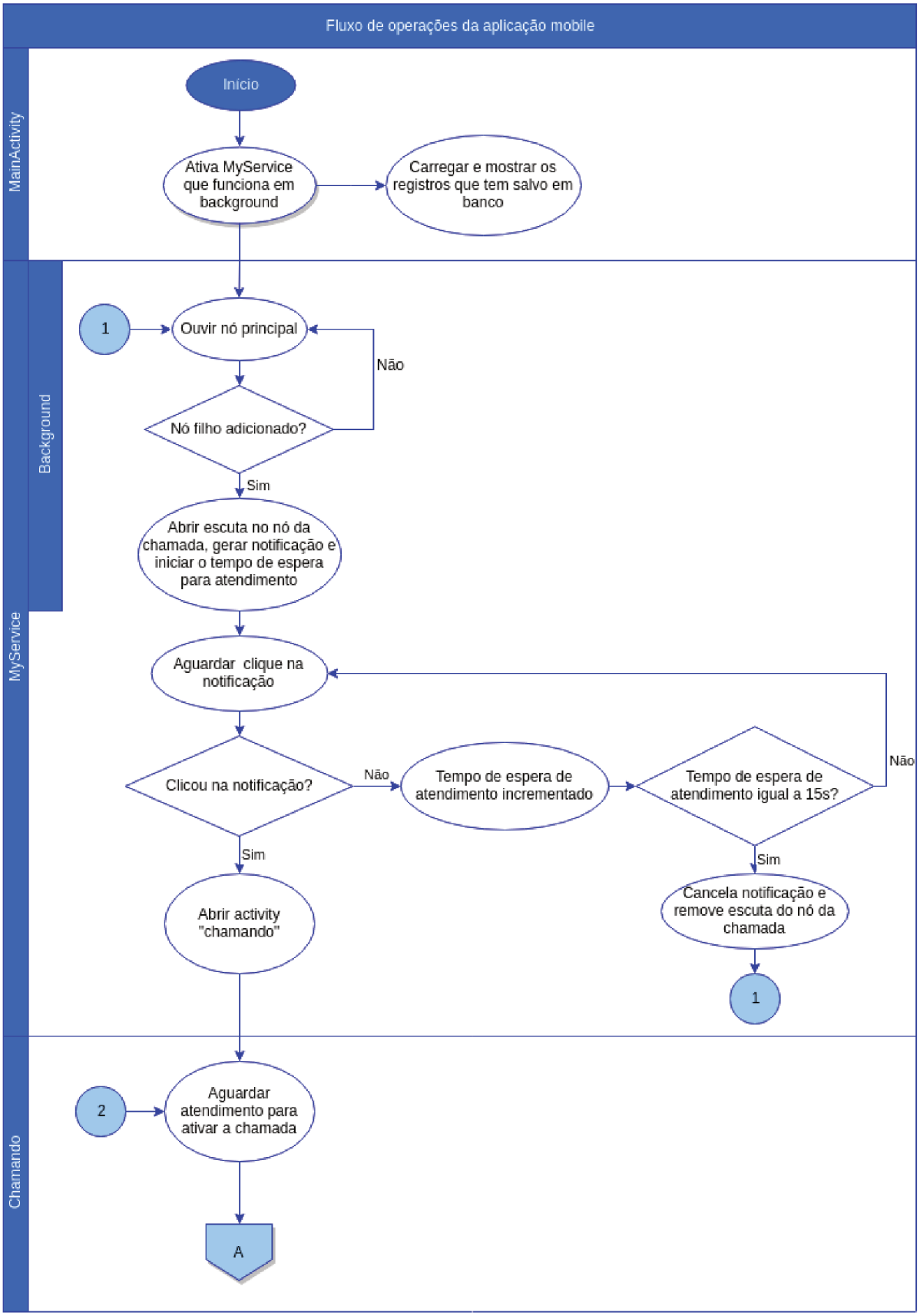


Figura 19 – Estrutura de pastas do projeto.

4.3.2 Fluxo de operações de aplicativo



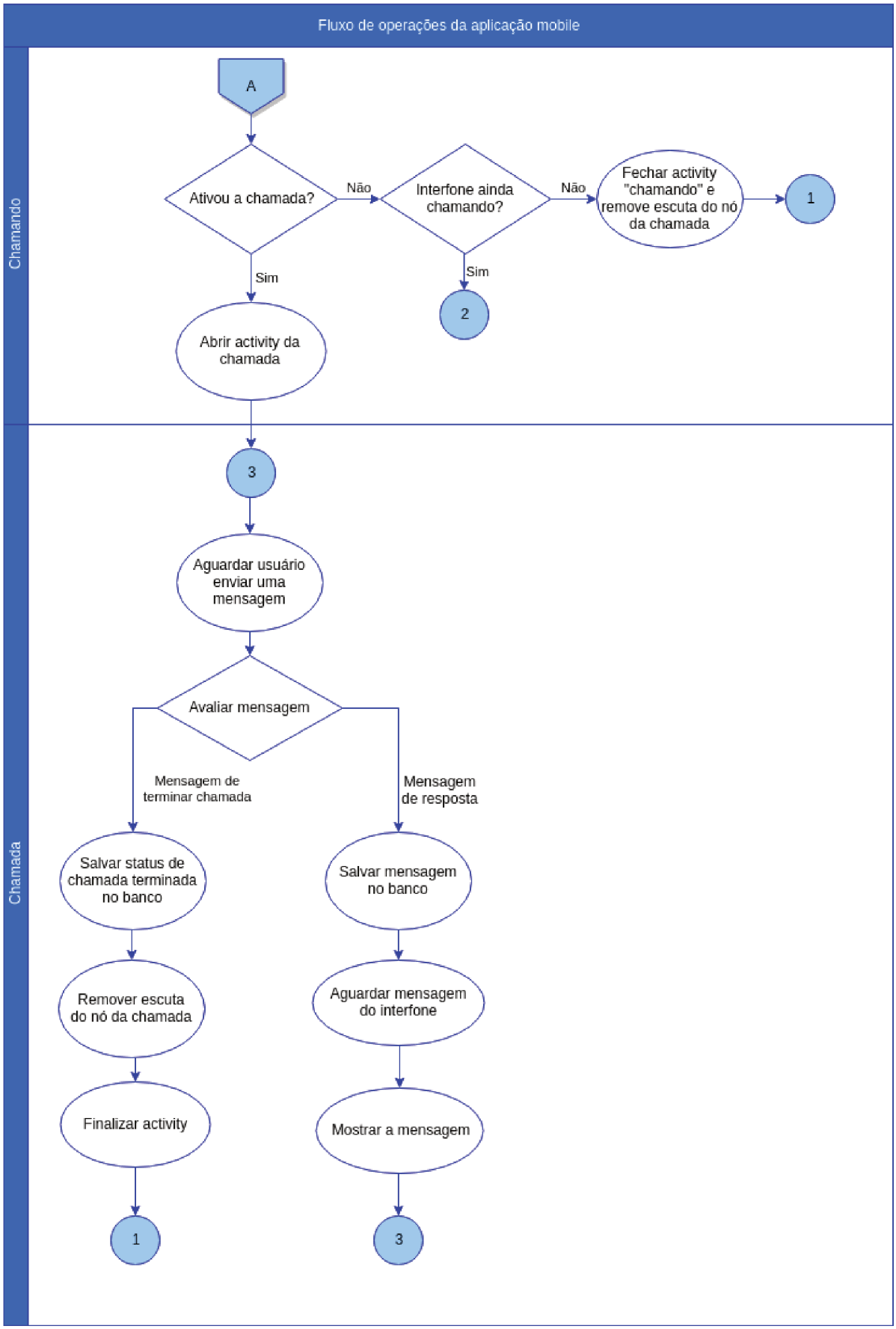


Figura 20 – Fluxo de operações da aplicação mobile.

Assim como no software desenvolvido para executar na Raspberry Pi, o aplicativo mobile também foi dividido por estados. O diagrama apresentado na figura 20 mostra o fluxo de operações que são executadas pelo aplicativo. É possível ver que cada divisão do fluxograma recebeu o nome das principais classes, pois são elas que fazem o fluxo de informações acontecerem.

Ao iniciar a aplicação, é apresentado a MainActivity que é responsável por ativar o serviço que executa em background e fazer o carregamento dos registros já existentes no banco. Além de poder ter acesso ao histórico de conversação visível pelo website do Firebase, o aplicativo exibe nesta tela inicial, todos os registros salvos em banco, contribuindo um pouco mais para gerar o conforto e segurança oferecidos pelo sistema. Ao fazer a abertura do aplicativo é iniciado em background a classe MyService, porém isto é necessário apenas uma vez pois, caso o usuário feche todos os aplicativos que executam em segundo plano, não será necessário abrir novamente a aplicação Interfone Virtual, pois, após a primeira iniciação, o próprio APP se responsabiliza de ficar sempre executando em segundo plano, garantindo que o usuário sempre esteja apto a receber os chamados.

A partir da iniciação da classe MyService, a aplicação passa a monitorar o nó principal do banco de dados Firebase com a intenção de identificar algum novo nó filho, que representará uma nova chamada de interfone. Caso seja adicionado um novo nó, é adicionado uma escuta específica para o novo nó e o aplicativo emite uma notificação com prioridade máxima, isso significa que o aplicativo tem o direito de acordar o dispositivo caso esteja dormindo, exibir o conteúdo da notificação na tela, não apenas quando se expande a barra de status, vibrar e emitir som. Foi definido esta prioridade pois uma ocorrência deste tipo tende a ser respondia no momento do chamado, portanto é necessário chamar a atenção de quem irá atender ao chamado de interfone. Em seguida o aplicativo inicia a contagem do tempo que irá permanecer chamando. Este tempo foi definido como 15 segundo assim como na Raspberry Pi, pois ambos são sincronizados. A partir de então é esperado o clique na notificação para abertura da Activity Chamando, caso não seja acionada a notificação durante o tempo de 15 segundos, a notificação será cancelada, será removido a escuta do nó da chamada específica e o aplicativo volta a funcionar apenas em background monitorando o banco de dados novamente.

Com o clique na notificação, é aberto a tela feita para o visitante ativar a chamada. Está Activity fica aberta até a chamada ser ativada ou até o interfone parar de tocar. Caso não seja ativada a chamada, a Activity Chamando é encerrada, sendo removida a escuta no nó específico da chamada e o aplicativo, assim como no estado anterior, volta a monitorar o nó principal em background.

Caso seja atendida a chamada, o aplicativo passa para fase da chamada ativa, Activity Chamada. Nesta fase é necessário que o aplicativo inicie a comunicação com uma mensagem de resposta ou pedindo o término da chamada. Caso o usuário decida terminar a chamada, é mudado o status do nó da chamada para terminado, a escuta dedicada para

esta conversação é removida e finalizada a Activity de chamada fazendo então o aplicativo voltar a monitorar o nó principal em background. Caso o usuário decida enviar alguma das mensagens pré-definidas, a escolhida será salva em banco e o aplicativo passará a aguardar pela resposta do visitantante. Quando a Raspberry Pi salvar em banco a resposta do visitante, será notificado ao aplicativo, que então mostrará a resposta para o morador e passará para fase da escolha da mensagem a ser enviada para o banco.

4.3.3 Componentes e telas de interação

Neste tópico serão apresentadas as interfaces desenvolvidas para aplicação, que buscam apresentar um modelo que seja o mais intuitivo possível, já que o sistema foi criado para atender diversos públicos.

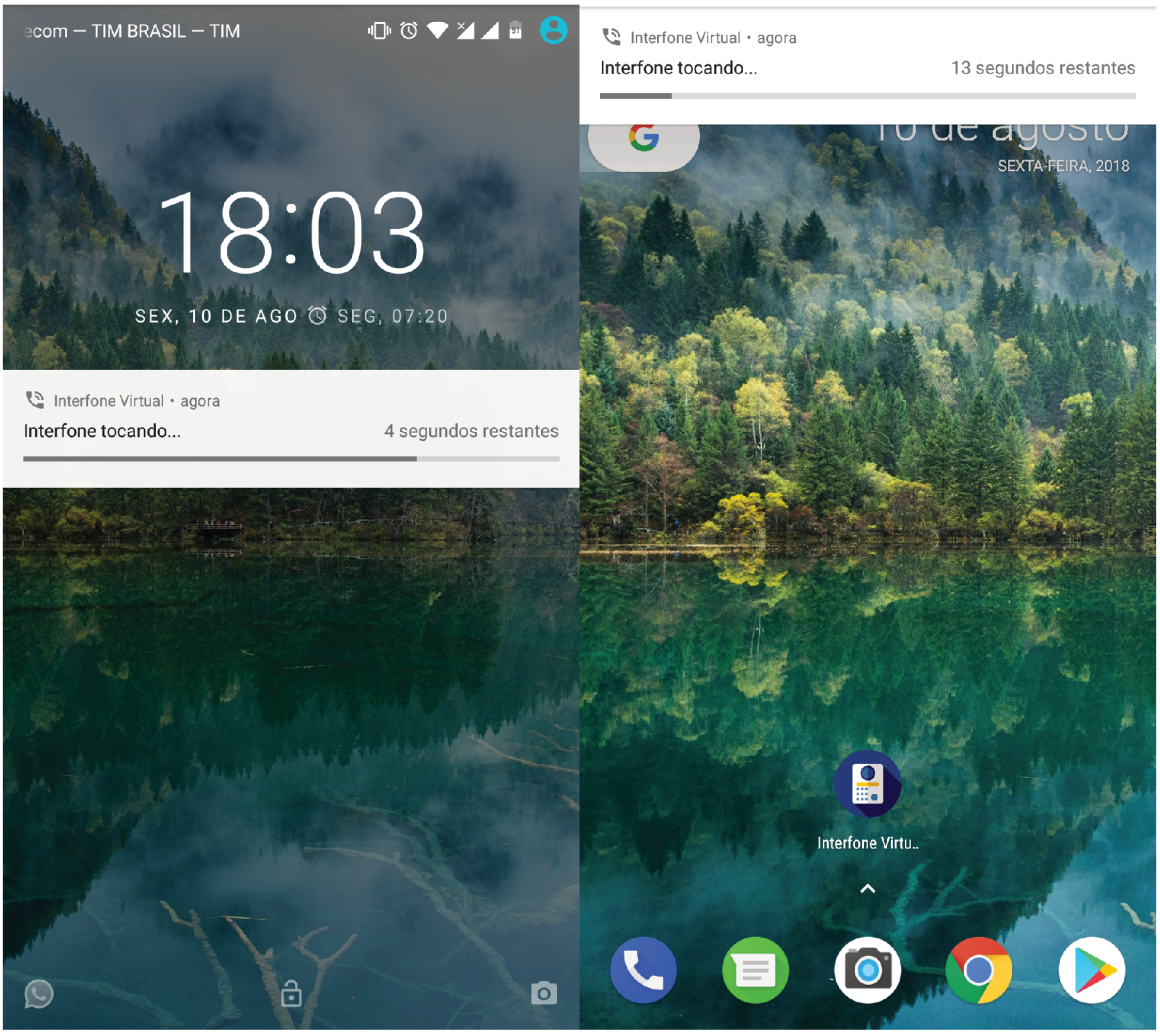


Figura 21 – Notificação do aplicativo Interfone Virtual.

Na figura 21 é possível ver a notificação que é emitida pelo aplicativo quando alguém toca o interfone. A primeira tela é referente a quando o aparelho está bloqueado, e a segunda imagem quando o aparelho está sendo usado normalmente pelo usuário. É

possível perceber pela imagem que a notificação conta com uma barra de progresso que é atualizada junto ao texto a direita dentro da notificação. Esses dois componentes da notificação mostram a contagem regressiva de quanto tempo resta para chamada parar de tocar. O ícone do aplicativo é visível na segunda tela.

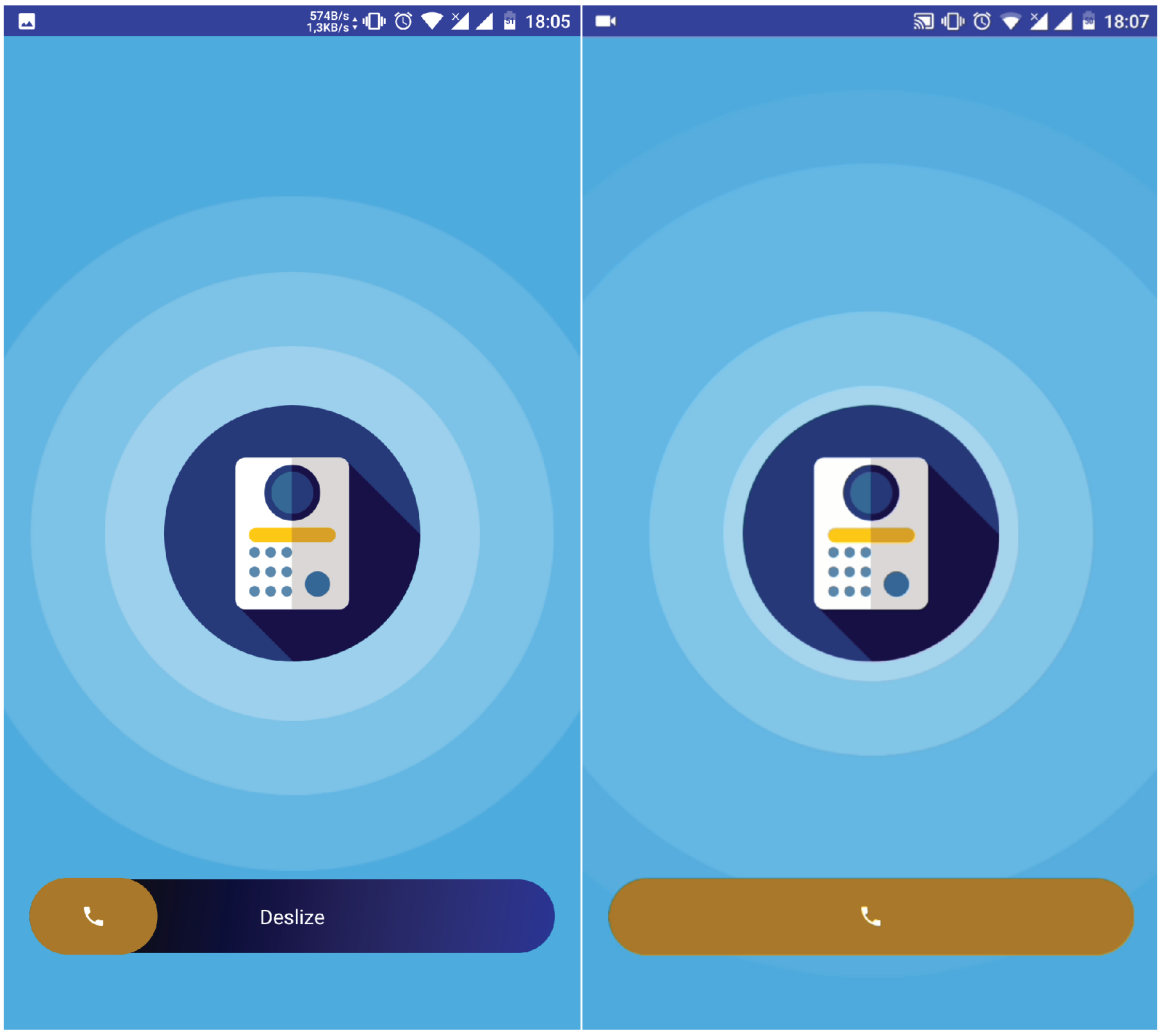


Figura 22 – Tela de ativação de chamada do aplicativo Interfone Virtual.

Na figura 22 é mostrado a tela referente a Activity utilizada para ativação da chamada. Nela é possível deslizar a parte dourada do botão da primeira tela até o final fazendo com que o botão se transforme no mostrado na segunda tela desta mesma figura ativando a chamada. Para construção deste botão foi utilizado uma biblioteca disponibilizada via GitHub a Swipe-Button[8] e para gerar o efeito de pulsação em torno do símbolo do interfone também foi utilizado outra biblioteca PulseView[9].

[8] Portal GitHub. Disponível em: <https://github.com/ebanx/swipe-button>. Acessado em 10/08/2018.

[9] Portal GitHub. Disponível em: <https://github.com/Devlight/PulseView>. Acessado em 10/08/2018.

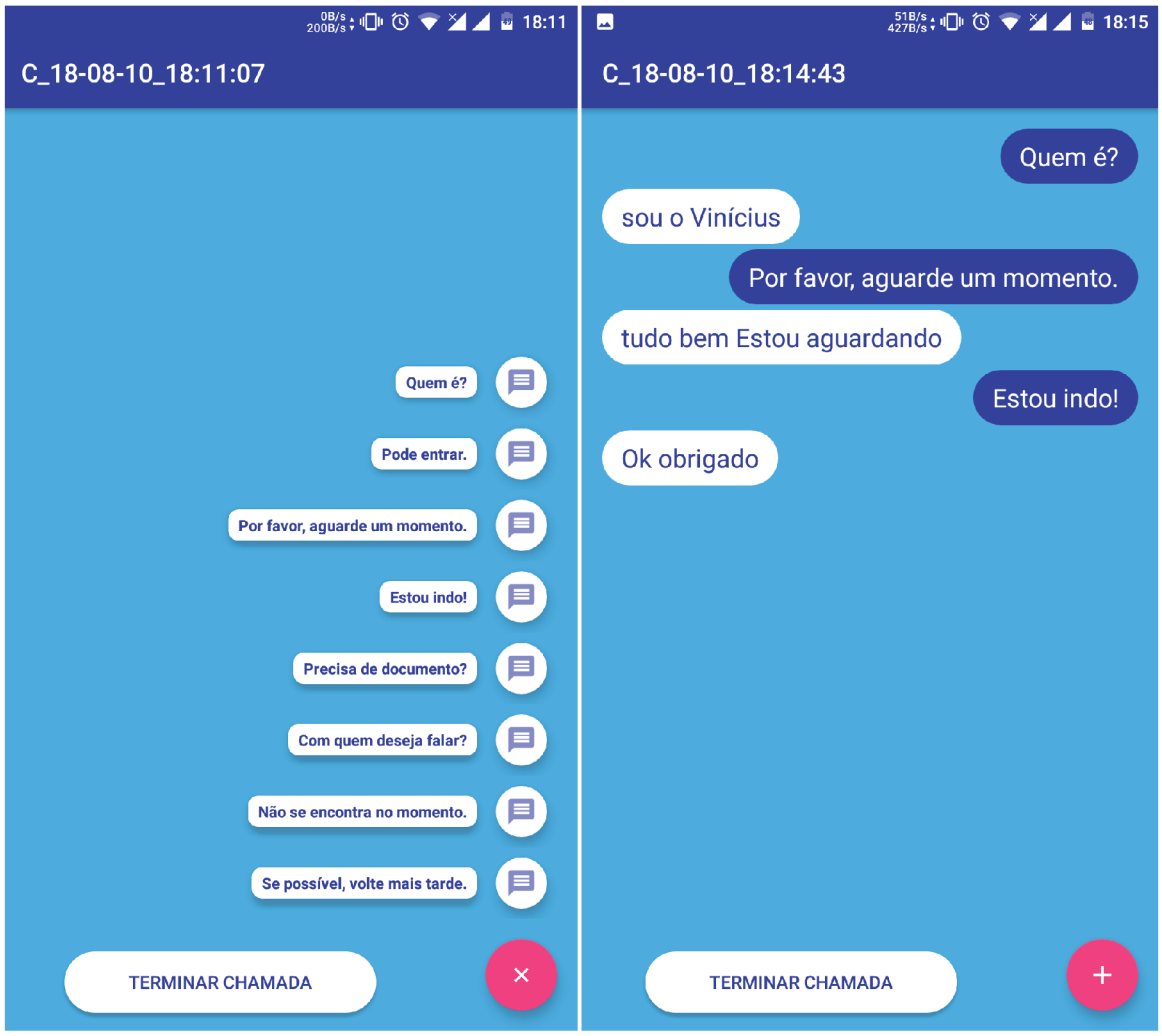


Figura 23 – Tela de bate papo do aplicativo Interfone Virtual.

A figura 23 apresenta a tela por onde o morador interage com o visitante. A primeira tela apresenta as mensagens de acesso rápido que são apresentadas ao apertar o botão que foi construído utilizando a biblioteca fab-speed-dial[10]. As mensagens enviadas pelo morador são apresentadas no texto em formato de bate papo com o fundo azul, e as enviadas pelo visitante com o fundo branco. Além das mensagens de conversação, há o botão que pode ser utilizado para terminar a chamada e que foi construído utilizando a biblioteca Progress Button Android[11].

As conversas salvas no banco de dados são também visualizadas por meio da tela inicial do aplicativo, apresentada na figura 24. Nesta tela é possível ver inicialmente todas as chamadas salvas, e expandindo uma chamada desejada é possível ver as conversas que aconteceram naquela chamada.

[10] Portal GitHub. Disponível em: <https://github.com/yavski/fab-speed-dial>. Acessado em 10/08/2018.

[11] Portal GitHub. Disponível em: <https://github.com/leandroBorgesFerreira/LoadingButtonAndroid>. Acessado em 10/08/2018.

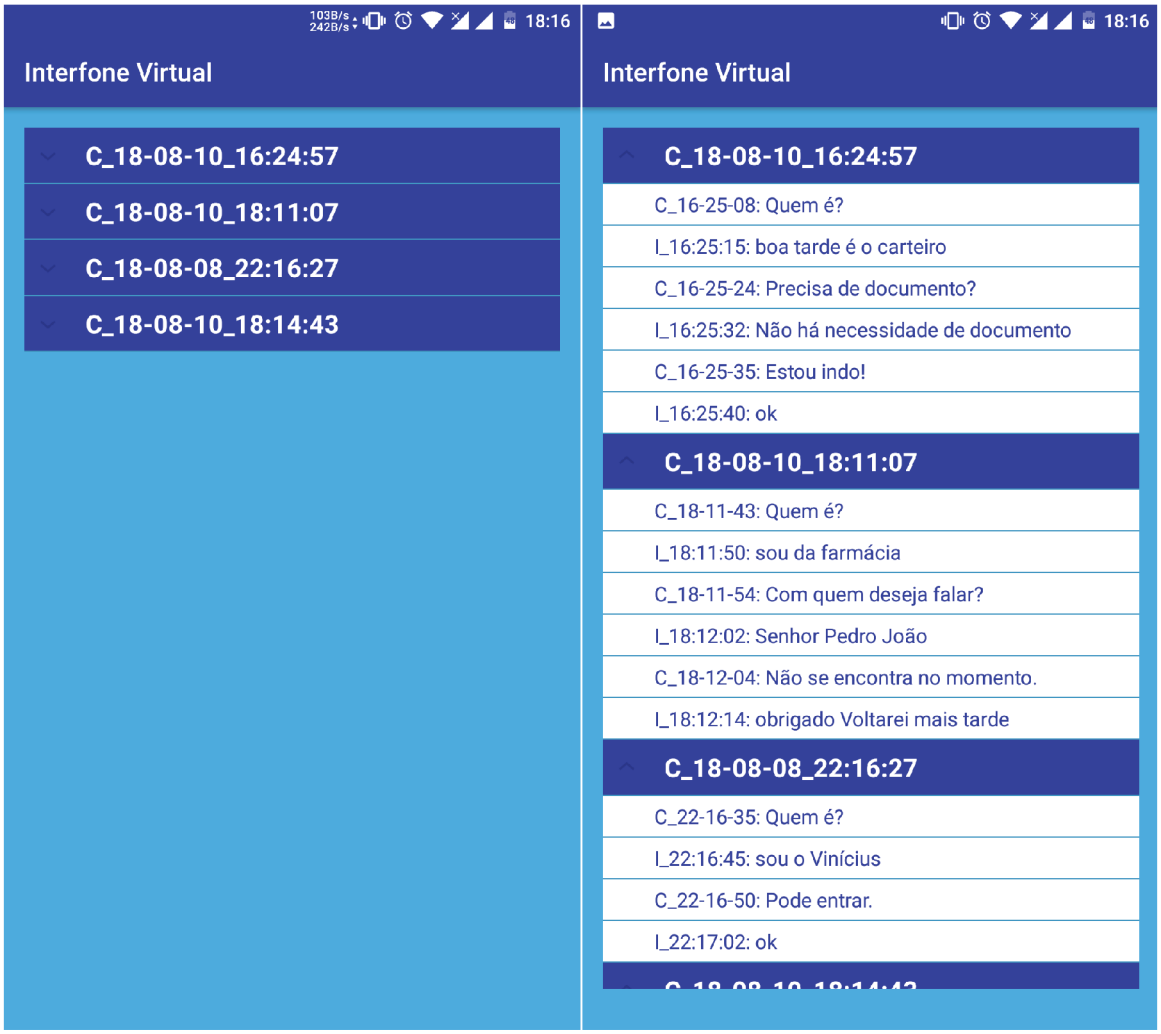


Figura 24 – Tela inicial do aplicativo Interfone Virtual.

Capítulo 5

Resultados e discussões

Com a evolução deste projeto, após muitas pesquisas pode-se observar o quanto o tema de automação cresceu absurdamente devido aos diversos benefícios trazidos pelo seu uso. E no meio deste universo de novidades, a automação voltada para as residências mostrou um forte impacto. Atualmente, existe várias soluções que se completam quando o assunto é automatizar uma casa, e o objetivo deste projeto foi poder contribuir com as tecnologias já existentes.

Buscando apresentar um novo complemento para automação residencial que seja de baixo custo e trazendo praticidade no seu uso e construção, pois os objetos aqui utilizados são de fácil acesso e há uma gama de dispositivos similares que podem ser substituídos por algum da preferência do comprador.

Por fazer o uso de ferramentas muito difundidas no mercado é de fácil acesso conseguir informação sobre determinado assunto. A Raspberry Pi como já foi mencionada, vem ganhando mercado a anos e com isso trazendo uma gama de informação voltada para praticamente todos os públicos, informação que vai desde o mais básico até o mais avançado, é possível achar tutoriais básicos para quem nunca teve contato com este mini computador, até fóruns de discussões sobre assuntos avançados. Todo esse conhecimento sendo criado por empresas, pessoas que tem o simples objetivo de ajudar os outros tirando dúvidas e até apresentando soluções prontas disponíveis para qualquer um. Um problema existente proveniente disso é o fato de que nem todas as informações fornecidas são verdadeiras, nem todas as ferramentas disponibilizadas fazem o que prometem ou que sua finalidade será realmente focada ao que o usuário procura. Portanto, quando foi definido o escopo deste projeto foi necessário muita pesquisa, atrás de ferramentas já existentes que poderiam ser úteis, e escolher as que mais se adequassem ao escopo definido, que é realmente fazer o trabalho de um engenheiro, analisar o que as ferramentas podem entregar e avaliar se é compensável utiliza-lás. Consequentemente para desenvolvimento dos softwares utilizados no sistema foi buscado diversas bibliotecas que proporcionaram fazer a integração dos diversos dispositivos utilizados, assim como métodos necessários para conversação com outros softwares.

Na Raspberry Pi duas bibliotecas foram de suma importância para a concretização do projeto, bibliotecas utilizadas para conversação com o Firebase e outra para reconhecimento da fala.

Como já mencionado o Firebase disponibiliza integração com Android, IOS e Web, porém neste projeto foi utilizado a linguagem Python que não tem uma biblioteca disponibilizada pelo próprio Firebase para conexão. Com isso foi necessário fazer o teste de bibliotecas disponibilizadas de forma gratuita, e que eram open source, pois, caso fosse necessário fazer alguma alteração no código da biblioteca, isso seria possível. Após recolher os requisitos que essa biblioteca deveria ter e visto que era necessário fazer a conexão ao banco de dados Firebase utilizando uma autenticação com email e senha, fazer consultas e inserções ao banco e principalmente, adicionar escutas ao banco, foi escolhida a biblioteca Pyrebase[1], que continha esses métodos e outros que poderiam ser utilizados numa possível continuação do projeto.

Para fazer a captação da voz e sua transformação em texto para gravação em banco, foi pesquisado uma biblioteca mais robusta já que neste caso há mais contratempos, como o fato de cada pessoa ter um tom de voz diferente, ter seu sotaque, o som ambiente atrapalhar, por isso foi necessário a busca de uma solução com uma confiabilidade maior. Para esta missão foi então escolhido a biblioteca SpeechRecognition[2] pelo fato de esta ser desenvolvida para linguagem Python e ter suporte a diversas APIs, o que possibilita a mudança para outra API caso fosse necessário.

A própria Raspberry Pi 3 Model B foi escolhida pelo fato de poder ser facilmente substituída por outro modelo dos diversos existentes, além de proporcionar a utilização de outra placa sem ser da família Raspberry, como por exemplo a Beaglebone, com as devidas configurações necessárias.

Os periféricos utilizados, como o push button poder ser substituído por diversos componentes que façam o acionamento do software ou o headset que também pode ser alterado pelo uso de um alto falante e microfone separados, de acordo com a preferência de quem fizer o uso do sistema.

O Firebase foi escolhido por três motivos que se destacaram, o alto número de bibliotecas disponíveis para integração com outras linguagens, o banco de dados em tempo real que possibilitou a troca de mensagens o mais rápido possível e sua fácil configuração.

Dentre os motivos que influenciaram a escolha do Android como sistema operacional para qual o aplicativo foi desenvolvido, foi o fato de este ser o sistema mais utilizado no mundo, como já foi mencionado acima. Para fazer o desenvolvimento do aplicativo é utilizado a linguagem Java, que já tinha sido trabalhada no decorrer da graduação, outro ponto que contribui para escolha deste sistema. A ideia do aplicativo era ser o mais

[1] Portal GitHub. Disponível em: <https://github.com/thisbejim/Pyrebase>. Acessado em 13/08/2018.

[2] SpeechRecognition. Disponível em: <https://pypi.org/project/SpeechRecognition/>. Acessado em 13/08/2018.

intuitivo possível buscando otimizar o espaço em tela pois, mesmo existindo smartphones com telas relativamente grandes, o APP foi desenvolvido para executar em celulares de diversos tamanhos de telas, incluindo as pequenas.

Com a implementação do projeto, conforme apresentado na figura 25, pode-se perceber que o objetivo proposto foi alcançado. Houve alguns atrasos que foram notados, como o tempo que o software leva para fazer a transformação da voz em texto, um tempo que é afetado diretamente pela qualidade da internet, já que a tradução não é feito offline, mas sim, online, por meio da biblioteca utilizada. Um segundo atraso que é percebido acontece no momento em que uma nova chamada é cadastrada no banco e leva um tempo até o smartphone ser notificado. Outro impasse observado é o tempo que leva para fazer a tradução de voz para texto quando está em um ambiente com muito barulho, o que dificulta no entender do que foi dito. Entretanto, mesmo com estes obstáculos encontrados, após diversos teste executados, foi possível analisar que o sistema atende ao que foi proposto, proporcionando com êxito a conversação entre o interfone e o smartphone do morador.

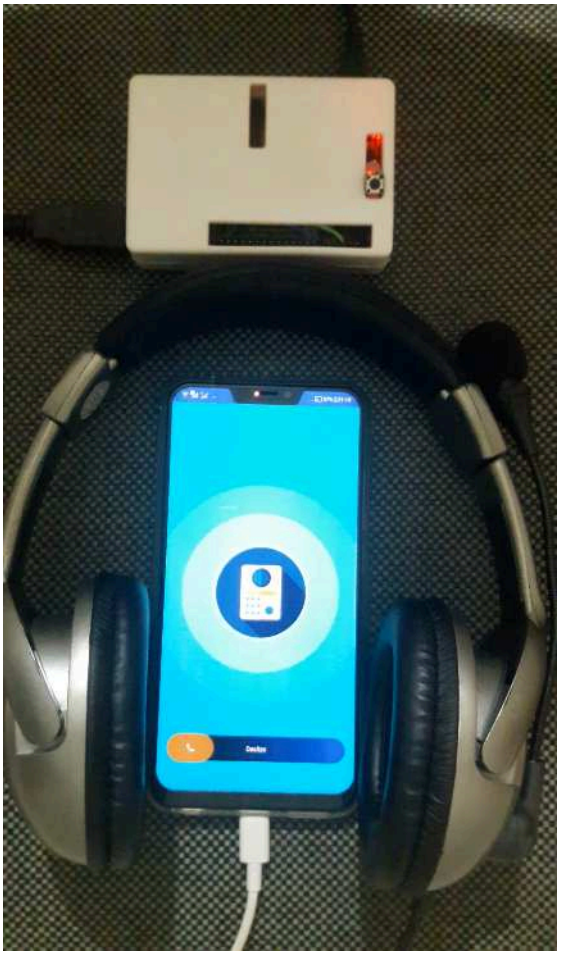


Figura 25 – Sistema montado.

Capítulo 6

Conclusão e considerações finais

Durante todo o tempo que foi dedicado para aprendizagem e construção deste projeto foi possível perceber o quão grande se tornou o número de ferramentas disponibilizadas para desenvolvimento de projetos voltados para automação residencial.

Depois de analisado diversas áreas de atuação deste segmento, descobrindo diversos sistema existentes para automação residencial e cada um com seu foco em especial, foi feita uma análise das ferramentas e encontrada uma que mais se adequa ao propósito deste sistema, que como já dito é buscar criar um sistema de reencaminhamento de chamadas de interfone para dispositivos móveis com um diferencial do mercado, pois foi encontrado sistemas com este propósito, porém fazendo o uso de uma central telefônica e o motivo deste sistema é utilizar puramente a internet.

Com a definição deste objetivo em mãos foram selecionadas as ferramentas que seriam utilizadas visando sempre um baixo custo, simplicidade e flexibilidade, buscando ferramentas que pudessem ser moldadas de acordo com que a necessidade fosse surgindo.

Com a conclusão do sistema pode-se perceber que o propósito foi cumprido, as ferramentas escolhidas além de proporcionarem suas funções eram de fácil reposição, até por apresentar ferramentas parecidas que também realizavam a mesma tarefa.

A Raspberry Pi foi um produto que se mostrou extremamente eficaz quando o assunto é automação residencial pelo fato de seu alto custo benefício, pelo fato de poder ser utilizado em diversos projetos e a flexibilidade que entrega ao usuário. Contudo, a Raspberry Pi pode apresentar algumas limitações que, dependendo do projeto podem implicar na necessidade da escolha de outro hardware, como por exemplo, o número de portas disponíveis ou ainda a velocidade de seu processador porque mesmo tendo uma velocidade de processamento que vem sendo melhorada constantemente, ainda há sistemas complexos que necessitam de uma velocidade superior. Todavia para aplicações mais simples que não requerem um nível tão alto de processamento, como a automação residencial, a Raspberry Pi se mostrou uma ótima ferramenta.

O Android Studio que foi a IDE utilizada para o desenvolvimento do aplicativo se mostrou extremamente eficiente, com o tempo, ao se ganhar mais afinidade com a plataforma,

ela se mostrou muito intuitiva e muito completa, uma IDE que pode ser utilizada para projetos simples de pequeno porte, até projetos empresariais muito complexos devido ao alto número de ferramentas que vêm embutidas no software. Por meio dela foi possível desenvolver com êxito a aplicação proposta, buscando simplicidade em seu layout, para que o aplicativo possa ser de fácil uso por pessoas de diferentes idades e fornecendo a comodidade e segurança que o sistema sugere.

Mesmo não tendo explorado nem 50% das ferramentas disponibilizadas pelo Firebase, ele se mostrou muito eficaz, promovendo um enorme ganho de tempo em um projeto, pois não há a necessidade de se desenvolver nenhum servidor web, ele fez tudo isso por conta própria, fornecendo diversos recursos extraordinários. As ferramentas utilizadas, o banco em tempo real e a autenticação, se mostraram muito práticas, de simples configuração e fácil usabilidade mostrando que o Firebase é uma ótima ferramenta para ser usada em projetos de grande complexidade.

Pode-se concluir, que com a finalização deste projeto foi possível trabalhar com diversas tecnologias como banco de dados, programação orientada a objetos, sistemas embarcados, diversas bibliotecas com diferentes finalidades. A aplicação de todas essas tecnologias proporcionou o aprendizado em diversas áreas que um engenheiro de computação pode atuar.

A conclusão deste trabalho criou um sistema de reencaminhamento de chamadas de interfone para dispositivos móveis que cumpriu com os objetivos propostos, com algumas limitações, porém de modo bem-sucedido.

Referências

CABRAL, M. M. A.; CAMPOS, A. L. P. de S. **Sistemas de Automação Residencial de Baixo Custo: Uma Realidade Possível**. 2008.

CANALTECH. **O que é headset?** 2018. Acessado em: 30 de jul. 2018. Disponível em: <https://canaltech.com.br/produtos/O-que-e-headset/>.

DENNIS, A. K. **Raspberry Pi home automation with Arduino: Automate your home with a set of exciting projects for the Raspberry Pi**. Birmingham: Packt Publishing Ltd, 2013.

DEVELOPER. **Arquitetura da plataforma**. 2018. Acessado em: 26 de jul. 2018. Disponível em: <https://developer.android.com/guide/platform/>.

FERREIRA, V. Z. G. **A Domótica Como Instrumento para a Melhoria da Qualidade de Vida dos portadores de Deficiência.** Dissertação (Monografia) — Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, João Pessoa, 2010.

GOMES, R. C.; FERNANDES, J. A. R.; FERREIRA, V. C. **Sistema operacional Android**. TCC (Graduação) - Curso de Engenharia de Telecomunicações — Universidade Federal Fluminense, Niterói, 2012.

OTONI, P. P. **Ambiente para automação via Web Semântica utilizando Linus embarcado em micro controladores ARM**. Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2013.

O'REILLY. **Connecting a Push Switch**. 2013. Acessado em: 02 de ago. 2018. Disponível em: <http://razzpisampler.oreilly.com/ch07.html>.

RASPBERRY, F. **Raspberry Pi Foundation**. 2018. Acessado em: 30 de jul. 2018. Disponível em: <https://www.raspberrypi.org/help/faqs/>.

RASPBIAN. **Raspbian**. 2018. Acessado em: 30 de jul. 2018. Disponível em: <https://www.raspberrypi.org/downloads/raspbian/>.

RUWAIDA, B.; MINKKINEN, T. **Home automation system: A cheap and open-source alternative to control household appliances.** Tese (Bacharelado) — School of Information and Communication Technology (ICT), KTH Royal Institute of Technology, Estocolmo, 2013.

TAPR. **Open Hardware License Version 1.0**. 2007. Acessado em: 24 de jul 2018. Disponível em: <http://www.tapr.org/TAPR_Open_Hardware_License_v1.0.txt>.

TECMUNDO. **A história do Android, o robô que domina o mercado mobile**. 2017. Acessado em: 25 de jul. 2018. Disponível em: <https://www.tecmundo.com.br/ciencia/120933-historia-android-robo-domina-o-mercado-mobile-video.htm>.