**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO**
**CENTRO TECNOLÓGICO**
**DEPARTAMENTO DE ENGENHARIA ELÉTRICA**
**PROJETO DE GRADUAÇÃO**

**RAFAEL CATABRIGA BASTIDA**

# MEDIÇÃO DE DESEMPENHO DE SMP UTILIZANDO FUNÇÕES DO ANDROID

VITÓRIA – ES
AGOSTO/2014

RAFAEL CATABRIGA BASTIDA

# MEDIÇÃO DE DESEMPENHO DE SMP UTILIZANDO FUNÇÕES DO ANDROID

Parte manuscrita do Projeto de Graduação do aluno **Rafael Catabriga Bastida**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Dr. Antônio Manuel Ferreira Frasson

VITÓRIA – ES
AGOSTO/2014

RAFAEL CATABRIGA BASTIDA

# MEDIÇÃO DE DESEMPENHO DE SMP UTILIZANDO FUNÇÕES DO ANDROID

Parte manuscrita do Projeto de Graduação do aluno **Rafael Catabriga Bastida**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovada em 19, de agosto de 2014.

**COMISSÃO EXAMINADORA:**

**Prof. Dr. Antônio Manuel Ferreira Frasson**
**Universidade Federal do Espírito Santo**
**Orientador**

**Prof. Dr. Anilton Salles Garcia**
**Universidade Federal do Espírito Santo**
**Examinador**

**Eng. José Luiz Borba**
**Universidade Federal do Espírito Santo**
**Examinador**

Agradeço ao meu pai Miguel e à minha mãe Dulcinéia pelo apoio e pelos esforços para chegar até este momento. Agradeço também a minha família e amigos que fizeram parte deste período. Muito obrigado aos professores da UFES, em especial aos professores do Departamento de Engenharia Elétrica, pelo conhecimento compartilhado. E também aos funcionários da Elétrica, Guta, Nei e Binho, que sempre estavam dispostos a nos ajudar nos momentos mais desesperadores do curso.

# RESUMO

Neste trabalho é apresentado um programa de monitoramento remoto para análise de serviço móvel pessoal utilizando um aparelho celular com sistema operacional Android. Esse programa permitirá que seja feita uma análise da qualidade do serviço de telefonia móvel prestado em determinado local ou região. O aplicativo Android desenvolvido permite a execução de teste da intensidade do sinal recebido pela célula à qual o móvel está conectado e teste de chamada, ambos para as tecnologias GSM e UMTS. O trabalho contém uma breve descrição dos indicadores fiscalizados pela Anatel e a explicação dos códigos do aplicativo e do programa de monitoramento em *shell script*. O aplicativo foi desenvolvido utilizando um ambiente integrado de desenvolvimento em Java juntamente com as ferramentas necessárias para criação de aplicativos Android e o programa em *shell script* criado em um computador com sistema operacional Ubuntu, além de utilizar o protocolo de aplicação SSH para comunicação entre o computador e o aparelho celular. O programa desenvolvido atingiu as expectativas esperadas dentro das limitações impostas pelo sistema operacional Android para execução de determinadas ações.

# LISTA DE FIGURAS

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| SMP | Serviço Móvel Pessoal |
| GSM | *Global System for Mobile Communications* |
| UMTS | *Universal Mobile Telecommunications System* |
| Anatel | Agência Nacional de Telecomunicações |
| ADT | *Android Development Tools* |
| SDK | *Software Development Kit* |
| SSH | *Secure Shell* |
| PMM | Período de Maior Movimento |
| PMT | Período de Maior Tráfego |
| LAC | *Location Area Code* |
| RAC | *Routing Area Code* |
| PSC | *Primary Scrambling Code* |
| MCC | *Mobile Country Code* |
| MNC | *Mobile Network Code* |
| LTE | *Long-Term Evolution* |

# SUMÁRIO

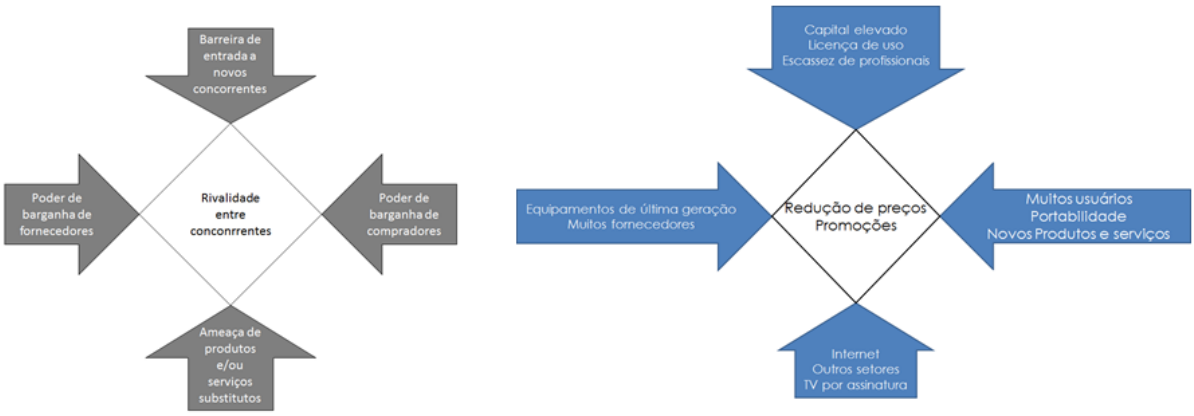
# 1 INTRODUÇÃO

A indústria de telefonia móvel movimenta uma quantidade enorme de capital todos os anos. Isso pode ser percebido pelos valores em vendas das principais empresas de telefonia móvel no Brasil. Em 2013, Claro, Oi, TIM e Vivo obtiveram vendas na casa de bilhões de dólares [1][2][3][4].

Além dos valores obtidos, a quantidade de usuários também é surpreendente. No Brasil, em maio de 2014, o número de usuários cadastrados foi de 275.451.832 [5]. Esse número é maior que a população no país, estimada em 202.980.000 no mês de agosto de 2014 [6]. O mesmo acontece para o estado do Espírito Santo, 4.501.841 de usuários e população de 3.885.000.

Esses números podem ser atraentes ao olhar de alguém interessado em entrar nesse mercado, porém a competitividade é muito elevada, conforme a análise feita utilizando as cinco forças de Poter [7][8], Figura 1. A barreira para novos concorrentes se deve ao elevado capital necessário, a obtenção de licença junto ao órgão regulamentador, à baixa disponibilidade de faixas de frequências para comunicações móveis (voz e dados) e a escassez de profissionais capacitados e experientes para gerenciamento; o poder de barganha dos fornecedores é enfraquecido devido ao grande número de fabricantes de equipamentos, o que minimiza o fato da necessidade de equipamentos de alta tecnologia; o poder de barganha dos compradores aumenta à medida que novos serviços e produtos são ofertados; a quantidade de substitutos cresce com o passar do tempo, principalmente pelo fato de ser cada vez mais comum o uso da internet como meio de comunicação por voz, o que torna empresas de outros segmentos, como TV por assinatura, potenciais substitutas; e a rivalidade entre concorrentes é destacada pelo enorme número de usuários, fazendo com que as empresas reduzam o preço de produtos e serviços, e consequentemente, o lucro, para atrair usuários de empresas concorrentes.

Figura 1 – As 5 forças de Porter




Fonte: Produção do próprio autor

Do ponto de vista do usuário, o que se deseja são serviços e produtos de qualidade e preços acessíveis. Uma forma avaliar este parâmetro é utilizando algum aplicativo que realize testes capazes de gerar informações e históricos sobre a qualidade desejada. Esses aplicativos disponíveis apenas permitem o teste in loco, não permitindo o monitoramento remoto. Como o próprio nome sugere, o monitoramento remoto permite que a verificação dos dados seja feita sem a presença do aparelho celular em mãos, permitindo o acompanhamento contínuo de determinado local.

Além disso, o monitoramento remoto é uma maneira mais eficiente do ponto de visto de uso de recursos profissionais. Não sendo necessário deslocar um profissional para determinado local toda vez que se necessita obter informações referentes ao funcionamento de determinada operadora de telefonia móvel. Basta apenas uma ida a determinado lugar e alocar o celular em local seguro. Após isso, os dados são obtidos sem a necessidade de uma pessoal no local. Vale destacar aqui como exemplo o caso do surgimento de um programa da Secretaria de Agricultura do Estado do Espírito Santo chamado Comunicação no Campo, projeto de autoria do Prof. Dr. Anilton Salles Garcia em que durante o período de estudo para implantação, se fez necessário a visita a 81 localidades no território do estado do Espírito Santo. Pode-se citar alguns gastos necessários para realizar tal estudo, deslocamento de equipe para cada localidade, custo por hora de cada profissional envolvido, caso seja observado uma coleta não satisfatória é necessária a visita à localidade novamente. Com uma ferramenta de monitoramento remoto a coleta pode ser feita por um longo período de tempo.

Outro ponto em que o uso de um monitoramento remoto se faz necessário é para o caso em que algum órgão do setor público contrata algum serviço de telefonia móvel de determinada operadora via processo de licitação. Citando o caso do Estado do Espírito Santo, caso alguma operadora ganhe o processo de licitação, hoje não é possível averiguar se o serviço prestado pela operadora realmente está de acordo com os termos do contrato, pois não existe nenhum modo de comprovar se o serviço está de acordo ou não. Os aplicativos disponíveis funcionam apenas de forma pontual, sendo muito trabalhoso a obtenção de dados por um longo período e, mesmo assim, podem ser questionados os resultados pelo operadora, alegando que o problema seja pontual. O monitoramento remoto irá possibilitar um uso contínuo de captura de dados e, desse modo, caso seja detectado a prestação de serviços fora do que foi contratado, gerar um relatório para notificação da prestação do serviço de baixa qualidade.

# 2 OBJETIVO

Este projeto possui como objetivo o desenvolvimento de uma ferramenta de monitoramento remoto de indicadores de funcionamento de um aparelho de telefonia móvel dotado de sistema operacional Android. Os indicadores analisados são nível de sinal recebido da célula a qual o móvel está conectado e teste de queda de chamadas. Estes indicadores são obtidos para as tecnologias GSM (*Global System for Mobile Communications*) e UMTS (*Universal Mobile Telecommunications System*).

Essa ferramenta de monitoramento remoto permitirá a análise dos dados obtidos em determinado local, sendo possível também a geração de uma base de dados para que seja apresentado em relatório, indicando o funcionamento do serviço de telefonia móvel ao longo de um período de tempo. Ao utilizar mais de um aparelho celular em pontos diferentes, será possível a análise de uma determinada região. Podendo ser utilizado para averiguação do serviço de telefonia móvel prestado por uma operadora de acordo com o instituído no processo de licitação.

# 3 METODOLOGIA

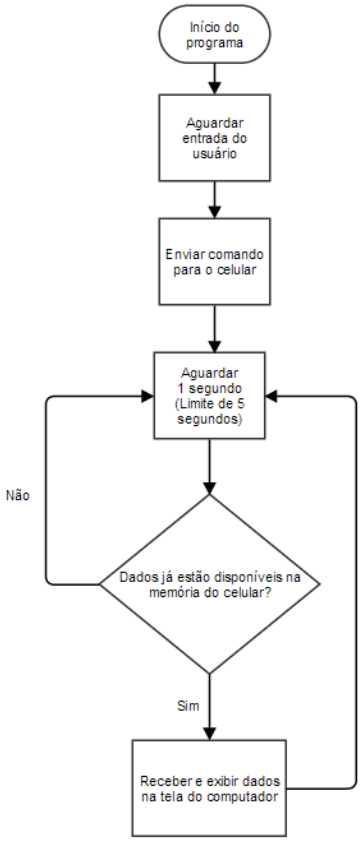
Antecedendo a parte de desenvolvimento do projeto, foi feito um estudo dos indicadores SMP (Serviço Móvel Pessoal) estipulados pela Anatel (Agência Nacional de Telecomunicações), identificando quais deles se direcionam para as necessidades deste projeto. As informações foram obtidas através do próprio site da Anatel[9].

A parte de desenvolvimento deste projeto ocorreu em duas partes tendo como base os fluxogramas exibidos nas Figuras 2 e 3: a criação de um aplicativo Android que fará a captura dos dados a serem monitorados e um código em *shell script* para fazer a comunicação entre computador e o celular e para exibição dos dados. Sendo o sistema operacional Android baseado em um kernel Linux, isso facilita uma comunicação entre um aparelho celular Android e um computador com sistema operacional Linux.

De acordo com a Figura 2, ao iniciar o programa no computador, é aguardado a entrada fornecida pelo usuário (uma tela dispondo as opções de teste é exibida); após recebido a informação do usuário, o programa envia para o celular um comando para que o aplicativo execute a ação requisitada pelo usuário; em seguida, o programa aguarda até que o aplicativo salve em determinado arquivo os dados obtidos pelo teste realizado, sendo estabelecido um limite de espera, para que o programa não espera indefinidamente os dados, pois se ocorrer a demora para obter os dados é uma indicação de que algum erro no aplicativo ocorreu; ao copiar este arquivo para o computador, o programa exibe para o usuário os dados obtidos. A partir daí, tem-se um ciclo de verificar se os dados do aplicativo já estão prontos e exibir estes dados para o usuário, até que seja interrompido pelo usuário.

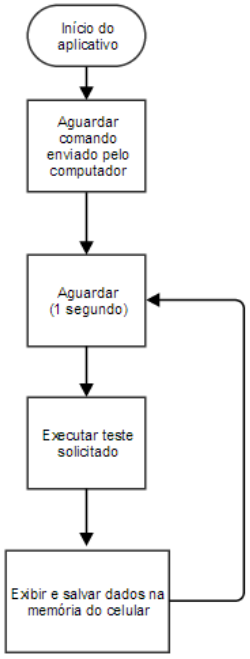
Da Figura 3, ao iniciar o aplicativo, este fica aguardando o recebimento do comando a ser executado enviado pelo computador; após obtido qual o comando referente ao teste solicitado, o aplicativo executa-o; em seguida, as informações obtidas são apresentadas na tela para o usuário e salvas em um arquivo para que o computador possa copiá-lo. Tem-se, então, um ciclo entre aguardar um determinado tempo, executar o teste e exibir e salva os dados.

Figura 2 – Fluxograma para o programa desenvolvido em *shell script*




Fonte: Produção do próprio autor

Figura 3 – Fluxograma para o aplicativo Android




Fonte: Produção do próprio autor

O aplicativo Android foi desenvolvido utilizando o Eclipse [10] como ambiente integrado de desenvolvimento em Java, obtido de forma gratuita diretamente do site de desenvolvimento do sistema operacional Android [11]. Junto ao Eclipse, obteve-se o Eclipse ADT (*Android Development Plugin*), um *plugin* utilizado para desenvolvimento de aplicativos Android, e as ferramentas do Android SDK (*Software Development Kit*).

O código em *shell* foi desenvolvido no sistema operacional Ubuntu 14.04 LTS [12]. Como *shell* não é uma linguagem de alto nível, os comandos são executados diretamente por um interpretador Unix.

A integração entre o aplicativo e o código em *shell* dá-se pelo uso de um aplicativo de servidor SSH (*Secure Shell*). Nesse projeto foi utilizado o SSHDroid [13]. Dessa forma, é possível acessar o aparelho celular Android pelo computador por linhas de comandos.

# 4 SERVIÇO MÓVEL PESSOAL

## 4.1 Anatel

A Anatel é uma das dez agências reguladoras existentes no Brasil. Com data de criação em 16 de julho de 1997 pela Lei Geral de Telecomunicações - Lei 9.472 [14]. Assim como as demais agências, sua função é fiscalizar a qualidades dos serviços prestados e estabelecer regras para as companhias privadas que prestam algum tipo de serviço público [15]. Vale destacar a independência administrativa e financeira da Anatel, não sendo subordinada a nenhum outro órgão nacional.

## 4.2 Definições

Os itens de verificação do SMP são utilizados para a fiscalização dos compromissos e metas estipulados pelo Regulamento de Gestão da Qualidade do Serviço Móvel Pessoal. De acordo com a Portaria no 445 de 3 de junho de 2014, existem 12 itens. De acordo com o site da Anatel, para a referida portaria, não havia descrição para SMP 10 e SMP 11. Esses indicadores são utilizados para todas tecnologias de telefonia móvel (GSM, UMTS e LTE).

Os indicadores descritos a seguir envolvem desde relações entre as operadoras e usuários (reclamação dos usuários, operação dos centros de atendimentos, tempo de respostas as solicitações dos usuários), relação entre as operadoras e a Anatel (reclamações diretamente à Anatel) e funcionamento dos serviços prestados pelas operadoras (quedas de chamadas, ligações completadas, envio de mensagens de texto, conexão de dados). Para este projeto tem-se interesse apenas nos indicadores de funcionamento do serviços, visto que esses envolvem o que é descrito ao longo do projeto.

O desacordo com os limites estipulados por esses indicadores sugerem que possa haver alguma falha com relação a abrangência da área de cobertura do sinal, irradiação do sinal pelas antenas da torres ou alguma singularidade devido ao ambiente de propagação do sinal (área densamente urbana, reflexão do sinal por barreiras, entre outros).

Tais indicadores serão utilizados para comparação com a análise feita após a captura. Isso indicará se o local ou região onde foram feitas as coletas de informação de funcionamento estão

de acordo com o mínimo que foi exigido pela Anatel. Caso esteja em desacordo, poderá ser apresentado um relatório à operadora solicitando medidas para correção do serviço de baixa qualidade.

### 4.2.1 SMP 1: Taxa de Reclamação

De acordo com [16], é o "indicador que avalia a relação entre o número total de reclamações recebidas pela prestadora, em todos os seus canais de atendimento (diretamente pelo usuário), e o número total de acessos em operação no mês."

### 4.2.2 SMP 2: Taxa de Reclamação na Anatel

De acordo com [16], é o "indicador que avalia a relação entre o número total de reclamações recebidas na Anatel, em desfavor da prestadora, e o número total de reclamações recebidas pela prestadora em todos os seus canais de atendimento, no mês."

### 4.2.3 SMP 3: Taxa de chamadas completadas para os centros de atendimento

De acordo com [16], é o "indicador que avalia a relação entre as tentativas e o completamento de todas as chamadas originadas na rede da prestadora e destinadas ao seu Centro de Atendimento, no mês. Essas chamadas devem ser completadas, em cada PMM, no mês, em, no mínimo, 95% dos casos."

### 4.2.4 SMP 4: Taxa de chamadas completadas

De acordo com [16], é o "indicador que avalia a relação entre o número total de chamadas originadas completadas e o número total de tentativas de originação de chamadas, contadas a partir da alocação do canal de voz em cada PMM, no mês, em, no mínimo, 67% dos casos."

### 4.2.5 SMP 5: Taxa de alocação de canal de tráfego

De acordo com [16], é o "indicador que avalia a relação entre as tentativas e os sucessos de acesso aos canais de tráfego. As tentativas de alocação de canal de tráfego devem ser concluídas com sucesso, em cada PMM, no mês, em, no mínimo, 95% dos casos."

### 4.2.6 SMP 6: Taxa de entrega de mensagem de texto

De acordo com [16], é o "indicador que avalia a relação entre o número total de tentativas de envio de Mensagens de Texto na rede da prestadora e o número total de Mensagens de Texto enviadas a partir da rede da prestadora e entregues ao usuário em até 60 segundos. Todas as tentativas de envio de Mensagens de Texto devem resultar em entrega ao usuário final em até 60 (sessenta) segundos em, no mínimo, 95% dos casos, no mês."

### 4.2.7 SMP 7: Taxa de queda de ligações

De acordo com [16], é o "indicador que avalia a relação entre o número total de chamadas completadas e o número total de chamadas interrompidas por queda de ligação. A quantidade de chamadas interrompidas por queda da ligação na rede da prestadora, em cada PMM, no mês, deve ser inferior a 2%."

### 4.2.8 SMP 8: Taxa de conexão de dados

De acordo com [16], é o "indicador que avalia a relação entre o número total de tentativas de conexão destinadas a conexões de dados e o número total de tentativas de conexão destinadas a conexões de dados estabelecidas. As tentativas de conexão destinadas a conexão de dados utilizando a rede do SMP, no PMT, devem ser estabelecidas em 98% dos casos, no mês."

### 4.2.9 SMP 9: Taxa de queda das conexões de dados

De acordo com [16], é o "indicador que avalia a relação entre o número total de tentativas de conexão destinadas a conexões de dados estabelecidas e o número total de quedas das conexões de dados. As tentativas de conexão destinadas a conexão de dados utilizando a rede do SMP, no PMT, devem ser estabelecidas em 98% dos casos, no mês."

### 4.2.10 SMP 12: Taxa de atendimento pela telefonista/atendente em sistema de autoatendimento

De acordo com [16], é o "indicador que avalia a relação entre o número total de tentativas de acesso às telefonistas/atendentes e o número total de chamadas atendidas pelas telefonistas/atendentes em até 20 segundos, nos Sistemas de Autoatendimento da prestadora. O tempo para o atendimento pela telefonista/atendente em Sistemas de Autoatendimento, quando esta opção for selecionada pelo usuário, deve ser de até 20 segundos, no mês, em, no mínimo, 90% dos casos."

### 4.2.11 SMP 13: Taxa de resposta ao usuário

De acordo com [16], é o "indicador que avalia o tempo de resposta, ao usuário, às solicitações de serviços ou pedidos de informação. Todas as solicitações de serviços ou pedidos de informação recebidos em qualquer Setor de Relacionamento, Atendimento e/ou Venda e Centros de Atendimento da prestadora, e que não possam ser respondidos ou efetivados de imediato, devem ser respondidos em até 5 dias úteis, em 95% dos casos, no mês."

### 4.2.12 SMP 14: Taxa de atendimento pessoal ao usuário

De acordo com [16], é o "Indicador que avalia o tempo de espera até o atendimento do usuário que, ao comparecer a qualquer Setor de Relacionamento, Atendimento e/ou Venda próprio da prestadora, deve ser atendido em até 30 minutos, em 95% dos casos, no mês."

# 5 PROGRAMA DE MONITORAMENTO REMOTO

Como citado anteriormente, existem alguns aplicativos que obtém informações sobre alguns dos indicadores de funcionamento de serviços de telefonia móvel. Porém, estes permitem apenas a visualização dessas informações diretamente no aparelho celular. A ausência de verificação das informações obtidas de forma remota é uma desvantagens desse aplicativos.

Com um aplicativo de monitoramento remoto pode-se obter as informações de determinado local de forma contínua e sem que necessite a presença de uma pessoal no local. Neste caso, basta apenas ir ao ponto onde se deseja fazer as medições e colocar o aparelho celular em um ambiente seguro e de modo que ele possa permanecer ligado por um longo período. Dessa forma, pode-se realizar o monitoramento não apenas de um local, mas também em vários pontos de uma região. Isso permitiria uma análise não apenas pontual, mas também de um determinado conjunto de pontos e, assim, obtendo uma análise mais abrangente.

Outras vantagens do uso de monitoramento remoto, como citado anteriormente, são redução de custos relacionados à obtenção de dados e desenvolvimento de processos de averiguação de funcionamento de serviços de telefonia móvel contratados por órgãos públicos via processos de licitação.

## 5.1 Aplicativo Android

O código para a criação do aplicativo foi desenvolvido de forma modular, onde cada parte do código foi separado em uma determinada função ou classe. Em seguida é descrito cada uma destas partes.

A Figura 4 apresenta o código no qual o aplicativo é iniciado. A primeira ação a ser tomada é ler o arquivo que o programa em *shell script* salvou na memória do celular referente a qual teste o aplicativo deve realizar (sinal recebido ou chamada de voz para GSM ou UMTS). Após obtida a string referente ao teste, uma seleção é feita para saber em qual contador de tempo o teste funcionará, ciclo de 2 segundos ou de 60 segundos.

Figura 4 – Parte do código referente a parte principal do aplicativo

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_neighboring_cell_information);

    //TextView Neighboring = (TextView)findViewById(R.id.test_1);

    Timer timer = new Timer();

    String s_techpara= fReadTechPara();

    if (s_techpara == "WC" || s_techpara == "GC"){

        timer.schedule(new TimerTask() {
            @Override
            public void run(){
                TimerThreadWC();
            }
        }, 1000, 60000);

    }else if (s_techpara == "WD"){

    }else {

        timer.schedule(new TimerTask() {
            @Override
            public void run(){
                TimerThread();
            }
        }, 0, 2000);
    }
}
```

Fonte: Baseado em código disponível em < http://steve.odyfamily.com/?p=12>. Acessado em: 29 de jun. de 2014

A Figura 5 apresenta a função a qual lê-se da memória do celular o arquivo que contém o teste a ser executado. A fim de garantir o correto funcionamento do código, a verificação é feita se no conteúdo do arquivo existe alguma das strings definidas para cada teste. A função retorna a string do teste a ser realizado.

Figura 5 – Função que lê do arquivo o código do teste a ser executado

```
public String fReadTechPara() {

    String var_r = "";

    String s_techpara = fReadFile("android_read.txt");

    if (s_techpara.contains("WV")) {
        var_r = "WV";
    } else if (s_techpara.contains("WC")) {
        var_r = "WC";
    } else if (s_techpara.contains("WD")) {
        var_r = "WD";
    } else if (s_techpara.contains("GV")) {
        var_r = "GV";
    } else if (s_techpara.contains("GC")) {
        var_r = "GC";
    }

    return var_r;
}
```

Fonte: Produção do próprio autor

Na Figura 6, tem-se a função responsável por ler determinado arquivo passado como parâmetro. O arquivo a ser lido deve estar contido num diretório específico, neste caso /sdcard/PG/. A leitura do conteúdo do arquivo é feita linha-a-linha e ao final é tornado o conteúdo do tipo string do mesmo modo como contido no arquivo.

Figura 6 – Função que lê arquivo da memória do celular

```
//**********************************************************************
//********** lê arquivo da memória externa ***************************
//**********************************************************************
public String fReadFile(String myFileText) {

    File sdcard = new File("/sdcard/PG/");

    File file = new File(sdcard,myFileText);

    StringBuilder text = new StringBuilder();

    try {
        BufferedReader br = new BufferedReader(new FileReader(file));
        String line;

        while ((line = br.readLine()) != null) {
            text.append(line);
            text.append('\n');
        }
    }
    catch (IOException e) {

    }

    String r = text.toString();
    return r;

}
```

Fonte: Baseado em código disponível em < http://stackoverflow.com/questions/12421814/how-can-i-read-a-text-file-in-android >. Acessado em: 06 de jul. de 2014

Na Figura 7, apresenta a thread dos contadores de tempo. Os parâmetros do contador, conforme Figura 4, são a thread do contador, o tempo em milissegundos para esperar pela primeira execução e o intervalo de tempo em milissegundos no qual será repetida a função. Essa thread não possui relação com a thread que contém os dados relacionados à interface gráfica do aplicativo. Desse modo, não é possível fazer qualquer alteração no conteúdo que será apresentado ao usuário. Para contornar isso, se fazer necessário chamar uma outra função que executará os comandos na thread da interface gráfica.

Figura 7 – Thread do contador de tempo

```
//*******************************************************
//********** Thread realacionada ao timer ***************
//********** Não possui permissão para modificar UI *****
//*******************************************************
private void TimerThread(){
    this.runOnUiThread(TimerThread_onUI);
}

private void TimerThreadWC(){
    this.runOnUiThread(TimerThread_onUI);
}
```

Fonte: Baseado em código disponível em < http://steve.odyfamily.com/?p=12>. Acessado em: 29 de jun. de 2014

A Figura 8 mostra a thread que executará as modificações na interface gráfica. Nela consta os comandos a serem executados pelo aplicativo para cada um dos testes selecionados pelo usuário no programa em *shell script*. Como está é uma thread chamada pelo contador de tempo para alterar a interface gráfica, não há ligação com as variáveis utilizadas na parte principal do aplicativo. Sendo assim, se faz necessário ler novamente o arquivo na memória do celular que contém qual teste a ser executado. Também são mostrados os comandos executados pelo aplicativo para o caso do teste de voz no UMTS. Após receber os dados da célula a qual o celular está conectado (descrito a seguir), é feita uma manipulação com os caracteres da string que contém as informações para que possa ser apresentado na tela do celular e também enviar ao computador. A Figura 9 apresenta a tela do aplicativo com os dados obtidos pelo teste de sinal UMTS. Nela consta o número de identificação da célula à qual o celular está conectado, este número possui 5 dígitos e é único para cada operadora de telefonia móvel; logo abaixo tem-se o nível de sinal da célula em dBm, para as tecnologias GSM e UMTS o limite que o aparelho consegue captar é -113dBm, sendo este valor referente ao sinal de nível muito baixo; em seguida tem-se o LAC (*Location Area Code*), um outro número de identificação referente a qual região a célula pertence – o LAC refere-se apenas para os recursos de voz, para dados usa-se o RAC (*Routing Area Code*) como identificador; por último, tem-se o PSC (*Primary Scrambling Code*), um outro identificador utilizado no processo de mudança dos canais de frequência para comunicação entre a célula e aparelho celular de forma segura.

Figura 8 – Thread que permite alterações na interface gráfico do aplicativo e executa o teste selecionado pelo usuário

```
//********** Método que funciona na mesma thread da UI ****************************
//********** Permitido modificar UI ***********************************************
//*************************************************************************************
private Runnable TimerThread_onUI = new Runnable() {
    public void run (){

        TextView Neighboring = (TextView)findViewById(R.id.test_1);
        TextView textTitle = (TextView)findViewById(R.id.title);

        String s_command = fReadTechPara();

        switch(s_command) {
            case "WV":
                String cWcdmaInfo = fCellInfo();

                String s_testTitle1 = "UMTS voice test";
                textTitle.setText(s_testTitle1);

                int index_cid = cWcdmaInfo.indexOf(" mCid=");
                int index_ss = cWcdmaInfo.indexOf("ss=");
                int index_lac = cWcdmaInfo.indexOf("mLac=");
                int index_psc = cWcdmaInfo.indexOf(" mPsc=");
                int index_endss = cWcdmaInfo.indexOf(" ber=");
                int index_endpsc = cWcdmaInfo.indexOf("} C");

                String var_Cid = cWcdmaInfo.substring(index_cid+6, index_psc);
                var_Cid = String.valueOf(Integer.parseInt(var_Cid) - 14876672);
                String var_SS = cWcdmaInfo.substring(index_ss+3, index_endss);
                var_SS = String.valueOf(2*(Integer.parseInt(var_SS))-113);
                String var_Lac = cWcdmaInfo.substring(index_lac+5, index_cid);
                String var_Psc = cWcdmaInfo.substring(index_psc+6, index_endpsc);

                String s_toPc = var_Cid + "\n" + var_SS + "dBm\n" + var_Lac + "\n" + var_Psc + "\n";
                String s_toApp = "Cell ID: " + var_Cid + "\nSignal Strengh: " + var_SS + "dBm\nLAC: " + var_Lac + "\nPSC: " + var_Psc + "\n";
                Neighboring.setText(s_toApp);
                fSaveFile(s_toPc);
                break;
```

Fonte: Produção do próprio autor

Figura 9 – Tela do aplicativo Android com o resultado do teste de sinal de voz UMTS


Fonte: Produção do próprio autor

A Figura 10 apresenta a função que é responsável por obter as informações da célula a qual o celular está conectado. O comando executado para obter essas informações também fornece dados sobre as células vizinhas as quais o celular também está recebendo sinal, conforme Figura 11. Para cada célula que o celular está recebendo sinal tem-se a indicação se esta é, ou não, a célula a qual ele está conectado (parâmetro mRegistered); outros números identificadores, como o identificador de país (MCC, *Mobile Country Code*) e da operadora (MNC, *Mobile Network Code*); tempo, em nano segundos, em que os dados foram obtidos desde que o sistema operacional foi iniciado. Para a aplicação deste projeto, não se fez necessário o uso dos dados das vizinhas, mas para obtê-los basta alterar o índice do comando Cinfolist.get( ). As informações aqui obtidas servem tanto para GSM ou UMTS, isso depende apenas de qual tecnologia o celular está configurado.

Figura 10 – Função que obtém informações da célula à qual o celular está conectado

```
//*****************************************************************************
//********** obtém informações de células ************************************
//*****************************************************************************
public String fCellInfo (){
    TelephonyManager telephonyManager = (TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);
    List<CellInfo> Cinfolist = telephonyManager.getAllCellInfo();

    CellInfo cCell = Cinfolist.get(0);
    String s_cCell = cCell.toString();

    return s_cCell;

}
```

Fonte: Baseado em código disponível em < http://developer.android.com/reference/android/telephony/CellInfo.html>. Acessado em: 21 de jun. de 2014

Figura 11 – Conteúdo completo da função que obtém as informações de células


Fonte: Produção do próprio autor

Após os dados serem obtidos e manipulados para enviar ao computador, faz-se necessário salvá-lo em um arquivo que será transferido para o computador. A função responsável por isso está apresentada na Figura 12. Ela usa como parâmetro a string que será salva no arquivo especificado dentro da função, neste caso, o arquivo android_file.txt no diretório /sdcard/PG/. Esta função também possui um alerta que aparecerá na tela do celular caso ocorra algum erro durante o processo de salvar o arquivo.

Figura 12 – Função que salva arquivo na memória do celular

```
//**********************************************************************
//********** salva arquivo na memória externa ************************
//**********************************************************************
public void fSaveFile(String myFileText) {
    try {
        File myFile = new File("/sdcard/PG/android_file.txt");
        myFile.createNewFile();
        FileOutputStream fOut = new FileOutputStream(myFile);
        OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);
        myOutWriter.append(myFileText);
        myOutWriter.close();
        fOut.close();

    } catch (Exception e) {
        Toast.makeText(getBaseContext(), e.getMessage(),
                Toast.LENGTH_SHORT).show();
    }

}
```

Fonte: Baseado em código disponível em < http://stackoverflow.com/questions/19853401/android-saving-to-sd-card-as-text-file>. Acessado em: 29 de jun. de 2014

Para o caso do teste de voz no GSM, o procedimento é semelhante ao do UMTS, conforme Figura 13. E a Figura 14 contém a tela do aplicativo para esse teste, sendo as informações contidas nela, a mesma para a tecnologia UMTS, com exceção do PSC, pois isto não foi implementado para o GSM.

Figura 13 – Opção de teste de sinal para GSM

```
case "GV":
    String cGsmInfo = fCellInfo();

    String s_testTitle3 = "GSM voice test";
    textTitle.setText(s_testTitle3);

    int index_gcid = cGsmInfo.indexOf(" mCid=");
    int index_gss = cGsmInfo.indexOf("ss=");
    int index_glac = cGsmInfo.indexOf(" mLac=");
    int index_gendcid = cGsmInfo.indexOf("} Cell");
    int index_gendss = cGsmInfo.indexOf(" ber=");

    String var_gCid = cGsmInfo.substring(index_gcid+6, index_gendcid);
    String var_gSS = cGsmInfo.substring(index_gss+3, index_gendss);
    var_gSS = String.valueOf(2*(Integer.parseInt(var_gSS))-113);
    String var_gLac = cGsmInfo.substring(index_glac+6, index_gcid);

    String s_gtoPc = var_gCid + "\n" + var_gSS + "dBm\n" + var_gLac + "\n";

    String s_gtoApp = "Cell ID: " + var_gCid + "\nSignal Strengh: " + var_gSS + "dBm\nLAC: " + var_gLac + "\n";

    Neighboring.setText(s_gtoApp);
    fSaveFile(s_gtoPc);

    break;
```

Fonte: Produção do próprio autor

Figura 14 - Tela do aplicativo Android com o resultado do teste de sinal de voz GSM


Fonte: Produção do próprio autor

A Figura 15 mostra os casos para testes de chamadas. Para isso é utilizada a função fMakeCall().

Figura 15 – Opção de teste de chamadas para GSM e UMTS

```
case "WC":
    String s_testTitle2 = "UMTS call test";
    textTitle.setText(s_testTitle2);

    fMakeCall();

    break;

case "GC":
    String s_testTitle4 = "GSM voice test";
    textTitle.setText(s_testTitle4);

    fMakeCall();

    break;
```

Fonte: Produção do próprio autor

A função fMakeCall( ), apresentada na Figura 16, invoca que o sistema Android realize uma chamada para o número de telefone contido na string “tell:9xxxxXXXX”.

Figura 16 – Função que executa as chamadas de voz

```
//*********************************************************************************
//********** realiza chamada  *****************************************************
//*********************************************************************************
public void fMakeCall (){
    PhoneCallListener phoneListener = new PhoneCallListener();
    TelephonyManager telephonyManager = (TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);
    telephonyManager.listen(phoneListener,PhoneStateListener.LISTEN_CALL_STATE);

    Intent callIntent = new Intent(Intent.ACTION_CALL);
    callIntent.setData(Uri.parse("tel:9xxxxXXXX"));
    startActivity(callIntent);

}
```

Fonte: Baseado em código disponível em < http://www.mkyong.com/android/how-to-make-a-phone-call-in-android/>. Acessado em: 27 de jul. de 2014

Para analisar os testes de chamadas é utilizada a classe contida na Figura 17. A qualquer alteração no estado que o celular se encontra (em espera, em chamada ou recebendo chamada) é salvo em um arquivo o estado do celular.

Figura 17 – Classe implementada para auxiliar no teste de chamadas

```
//**********************************************************************
//********** classe para salvar informações de chamada ******************
//**********************************************************************

  private class PhoneCallListener extends PhoneStateListener {

      String LOG_TAG = "LOGGING 123";

      @Override
      public void onCallStateChanged(int state, String incomingNumber) {

          if (TelephonyManager.CALL_STATE_RINGING == state) {
              // phone ringing
              Log.i(LOG_TAG, "RINGING, number: " + incomingNumber);
              fSaveFile("RINGING\n");
          }

          if (TelephonyManager.CALL_STATE_OFFHOOK == state) {
              // active
              Log.i(LOG_TAG, "OFFHOOK");
              fSaveFile("OFFHOOK\n");

          }

          if (TelephonyManager.CALL_STATE_IDLE == state) {
              // run when class initial and phone call ended,
              // need detect flag from CALL_STATE_OFFHOOK
              Log.i(LOG_TAG, "IDLE");
              fSaveFile("IDLE\n");

          }
      }
  }
```

Fonte: Baseado em código disponível em < http://www.mkyong.com/android/how-to-make-a-phone-call-in-android/>. Acessado em: 27 de jul. de 2014

Para que o aplicativo descrito anteriormente pudesse funcionar, foi necessário declarar algumas permissões de execução (leitura e escrita de arquivos, estado do celular etc.), pois algumas dessas ações alteram o conteúdo do celular, conforme Figura 18.

Figura 18 – Declaração das permissões para execução do aplicativo

```
<uses-sdk
    android:minSdkVersion="17"
    android:targetSdkVersion="19" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERACT_ACROSS_USERS_FULL" />
```

Fonte: Produção do próprio autor

## 5.2 Programa em *shell script*

O código *shell script* é responsável por fazer a interface entre o usuário e o computador e também entre o computador e o aplicativo. Para interface entre computador e aplicativo utiliza-se o protocolo de camada de aplicação SSH [17], utilizado para comunicação entre computares e execução de comandos de forma remota. Conforme citado anteriormente, o uso do protocolo SSH dá-se pelo uso do aplicativo SSHDroid.

Ao iniciar o código, Figura 19, é apresentado ao usuário uma tela de seleção de qual teste deve ser executado, teste de sinal para GSM/UMTS ou chamada de voz GSM/UMTS. A escolha é feita utilizando o teclado. Após captada a tecla apertada e verificada se está é uma das opções permitidas, define-se o conteúdo de uma variável de controle para indicar qual teste a ser realizado.

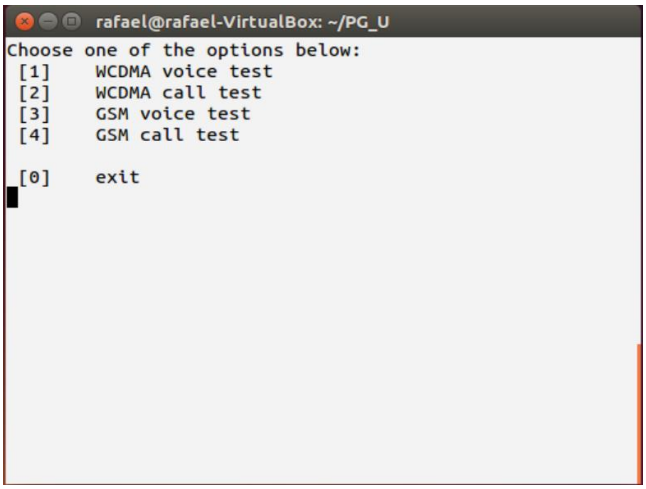
Figura 19 – Início do código em *shell script* e da tela de seleção de teste

```
#******************************
#**** inicio do codigo *******
#******************************

#escolher tecnologia
echo -en "Choose one of the options below:\n [1]\tWCDMA voice test\n [2]\tWCDMA call test\n"
echo -en " [3]\tGSM voice test\n [4]\tGSM call test\n\n [0]\texit\n"

sshpass -p $pcode ssh -p $pport root@$paddress cp /sdcard/PG/andr_in.txt /sdcard/PG/android_read.txt

#pegar resposta teclado
var_tech=""
flag_tech=0

while [ $flag_tech == 0 ]
do
    read -n 1 var_tech
    case $var_tech in
        "1") flag_tech=1
            var_tp="WV"
            clear
            echo -en "WCDMA voice test\n\n";;
        "2") flag_tech=1
            var_tp="WC"
            clear
            echo -en "WCDMA call test\n\n";;
        "3") flag_tech=1
            var_tp="GV"
            clear
            echo -en "GSM voice test\n\n";;
        "4") flag_tech=1
            var_tp="GC"
            clear
            echo -en "GSM call test\n\n";;
        "0") clear; exit;;
    esac
    echo -e -n "\033[1D"
done
```

Fonte: Produção do próprio autor

A Figura 20 mostra a tela de seleção implementada pelo código mostrado anteriormente.

Figura 20 – Tela exibida ao usuário para seleção de teste




Fonte: Produção do próprio autor

Na Figura 21, tem a verificação do conteúdo da variável de controle do teste a ser realizado. De acordo com a opção é modificado o conteúdo do arquivo que o aplicativo fará a leitura. A modificação ocorre utilizando a ferramenta sshpass [18]. Como a comunicação via SSH requer o uso a autenticação entre as partes, o sshpass permite que em um único comando seja feita a autenticação e o comando a ser executado.

Figura 21 – Seleção de código para execução de teste

```
case $var_tp in
    "GV") sshpass -p $pcode ssh -p $pport root@$paddress cp /sdcard/PG/GV.txt /sdcard/PG/android_read.txt
    fCallGV;; #chama funcao GSM voz

    "GC") sshpass -p $pcode ssh -p $pport root@$paddress cp /sdcard/PG/GC.txt /sdcard/PG/android_read.txt
    fCallWC;; #chama funcao GSM chamada

    "WV") sshpass -p $pcode ssh -p $pport root@$paddress cp /sdcard/PG/WV.txt /sdcard/PG/android_read.txt
    fCallWV;; #chama funcao WCDMA voz

    "WC") sshpass -p $pcode ssh -p $pport root@$paddress cp /sdcard/PG/WC.txt /sdcard/PG/android_read.txt
    fCallWC;; #chama funcao WCDMA chamada

esac
```

Fonte: Produção do próprio autor

Após a alteração do conteúdo do arquivo no celular, o aplicativo já inicia a ação necessária. E a cada ciclo é salvo o resultado em arquivo no celular para que possa ser transferido para o computador.

Para o teste de sinal GSM, Figura 22, o programa em *shell script* entra em um loop no qual copia o arquivo do celular para o computador, lê o conteúdo desse arquivo e exibe para o usuário o resultado obtido. A leitura do arquivo, assim como no aplicativo, é feita linha-a-linha. Tendo em cada linha um dos parâmetros do teste.

Figura 22 – Função que exibe informações do teste de sinal para GSM

```
fCallGV(){
    flag_end=1
    while [ $flag_end == 1 ]
    do
        sshpass -p $pcode ssh -p $pport root@$paddress cat /sdcard/PG/android_file.txt > /home/rafael/PG_U/pc_read.txt
        var_line=1

        while read pc_file
        do
            case $var_line in
            "1") var_CellID=$pc_file ;;
            "2") var_SS=$pc_file ;;
            "3") var_LAC=$pc_file ;;
            # "4") var_PSC=$pc_file ;;
            esac

            ((var_line++))
        done < "pc_read.txt"

        echo -en "\033[sCell Id: $var_CellID   Signal Strenght: $var_SS     \nLAC: $var_LAC \033[u"
        sleep $c_seg

    done

}
```

Fonte: Produção do próprio autor

Na Figura 23, tem-se o resultado obtido para um teste de sinal GSM. Os dados exibidos nessa tela são os mesmos dados que o aplicativo exibe no celular. Para o GSM tem-se o número de identificação da célula conectado, o nível de sinal e o LAC dela. A atualização do valor na tela é feita a cada segundo.

Figura 23 – Tela com resultados do teste de sinal de voz GSM


Fonte: Produção do próprio autor

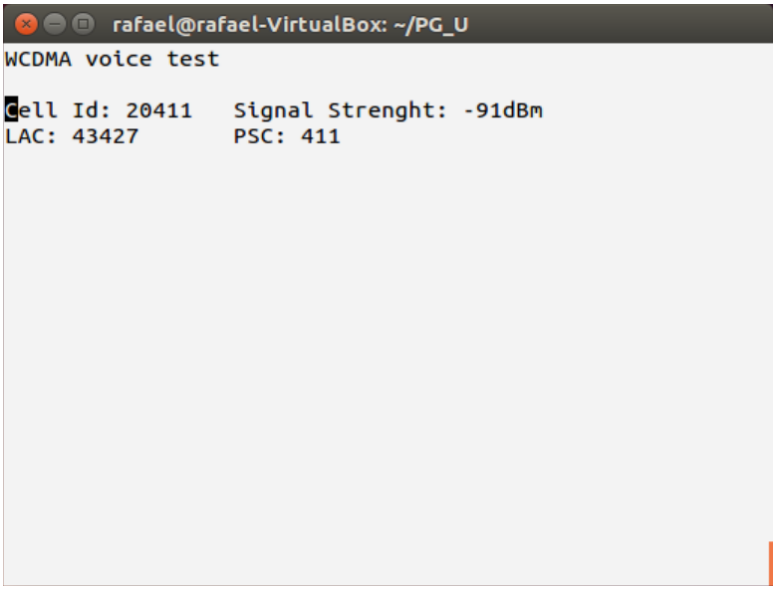
A mesma lógica descrita anteriormente é aplicada para o teste de sinal UMTS. Também com atualização da tela a cada segundo. O código é apresentado na Figura 24 e a tela de exibição na Figura 25, da mesma forma que no GSM, as informações exibidas na figura 25 são as mesmas que são exibidas na tela do celular.

Figura 24 - Função que exibe informações do teste de sinal para UMTS

```
fCallWV(){
    flag_end=1
    while [ $flag_end == 1 ]
    do
        sshpass -p $pcode ssh -p $pport root@$paddress cat /sdcard/PG/android_file.txt > /home/rafael/PG_U/pc_read.txt
        var_line=1

        while read pc_file
        do
            case $var_line in
            "1") var_CellID=$pc_file ;;
            "2") var_SS=$pc_file ;;
            "3") var_LAC=$pc_file ;;
            "4") var_PSC=$pc_file ;;
            esac

            ((var_line++))
        done < "pc_read.txt"

        echo -en "\033[sCell Id: $var_CellID   Signal Strenght: $var_SS     \nLAC: $var_LAC       PSC: $var_PSC \033[u"
        sleep $c_seg
        #((v_a++))

    done

}
```

Fonte: Produção do próprio autor

Figura 25 - Tela com resultados do teste de sinal de voz UMTS




Fonte: Produção do próprio autor

Para os teste de chamada de voz não há distinção entre as tecnologias, visto que o procedimento executado pelo celular não muda. Sendo assim, o mesmo código é utilizado para os teste em GSM e UMTS, Figura 26.

Ao ser escolhida a opção de teste de chamada, o programa em *shell script* entra em um loop no qual copiará o arquivo do celular para o computador e analisará o conteúdo deste arquivo. Inicialmente, o estado do celular é “em espera” (IDLE). Ao iniciar a chamada o estado muda para “em chamada” (OFFHOOK). Caso a chamada termine, o estado do celular volta para “em espera”. Baseado nessa mudança de estados é feito o teste de chamada. Um período de tempo é determinado para a duração da chamada. Caso a chamada termine antes de período, sem a ação de algum dos dois celulares, é considerada uma queda de chamada. E se a ligação ocorre até o tempo definido, é considerada uma chamada bem-sucedida. De acordo com o tipo de chamada, é atualizado o contador para cada um dos tipos.

Figura 26 - Função que exibe informações do teste de chamada

```
fCallWC(){

    flag_end=1
    var_cseg=0
    var_calldrop=0
    var_callgood=0

    while [ $flag_end == 1 ]
    do
        sshpass -p $pcode ssh -p $pport root@$paddress cat /sdcard/PG/android_file.txt > /home/rafael/PG_U/pc_read.txt
        while read pc_file
        do
            var_state=$pc_file

        done < "pc_read.txt"

        echo -en "\033[sCall State: $var_state      \nDropped Call: $var_calldrop     \nSuccessful Call: $var_callgood     \033[u"

        if [ $var_state == "OFFHOOK" ]
        then
            ((var_cseg++))
        else
            if [ $var_cseg != 0 ]
            then
                if [ $var_cseg -lt 30 ]
                then
                    ((var_calldrop++))
                else
                    ((var_callgood++))
                fi
                var_cseg=0
            fi
        fi

        sleep $c_seg
    done
}
```

Fonte: Produção do próprio autor

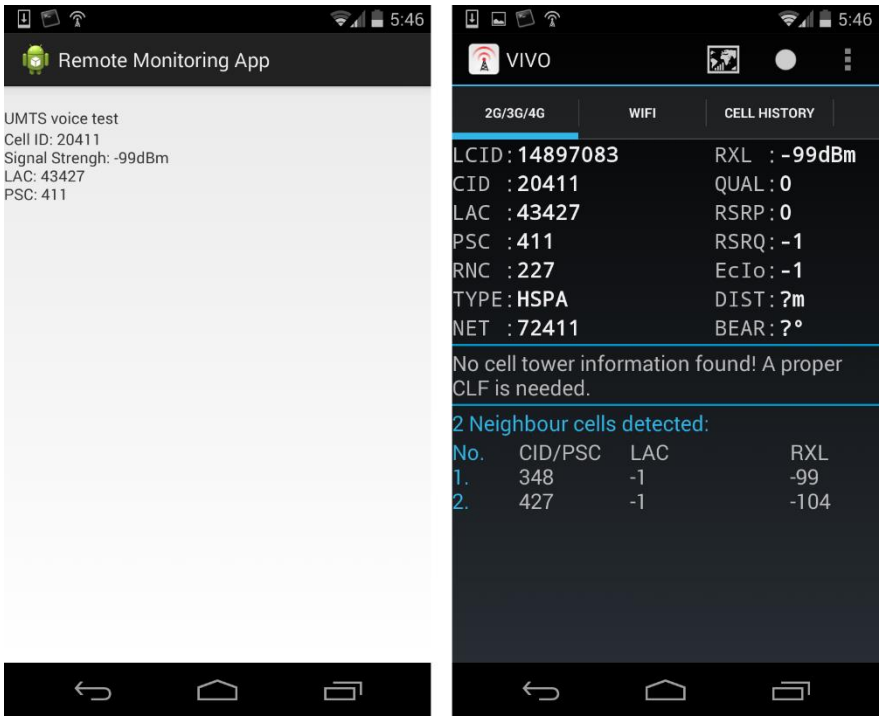
# 6 CONCLUSÕES

Neste trabalho foi apresentado o desenvolvimento do programa para monitoramento remoto de indicadores de funcionamento de um aparelho de telefonia móvel, sendo explicitadas as etapas do projeto.

Após a finalização do desenvolvimento dos códigos para o aplicativo e para o programa em *shell script*, pode-se validar os dados obtidos comparando-os com outros aplicativos disponíveis que executam tais capturas de dados. Neste caso, foi utilizado o G-MoN [19].

A Figura 27 mostra a comparação entre os dados obtidos pelo aplicativo desenvolvido e pelo G-MoN.

Figura 27 – Comparação entre aplicativo desenvolvido e outro já existente




Fonte: Produção do próprio autor

O funcionamento da ferramenta de monitoramento mostra que é possível a obtenção de dados referentes aos serviços de telefonia móvel para análise do mesmo em determinado local. Além disso, mostra-se que é possível também para análise de uma região, sendo necessário para isso o uso de mais de um aparelho celular alocados em diferentes pontos.

Sendo assim, mostrou-se que o objetivo de facilitar os testes de telefonia foi alcançado. Comprovando que não se torna necessário a presença de uma pessoa no local de teste. Isso se reflete em uma redução de custos para empresa ou entidade responsável em realizar tais testes, pois não será necessário enviar uma pessoa toda vez que necessário (isso inclui custos da carga horária do funcionário, gasto com transporte e outros).

Um problema encontrado durante o desenvolvimento do programa foi a limitação imposta pelo sistema operacional Android que não permite acesso a todas as funcionalidades através da execução de comandos por linhas de códigos. Isso acabou afetando a liberdade de uso de acordo com as necessidades do programa. Dentre as limitações estão a mudança de tecnologia de uso do aparelho (não é possível mudar via código de GSM para UMTS e vice-versa); executar via código outros aplicativos já instalados, como o SSHDroid; e alternar via código entre transferências de dados via WiFi e conexão de dados. Como consequência, os aplicativos necessários devem ser inicializados logo ao ligar o aparelho celular, para que não afete a execução do programa de monitoramento. Essas limitações já são discutidas há tempos por desenvolvedores de aplicativos Android [20],[21],[22].

Dentre as sugestões propostas em [21], está o desenvolvimento de um firmware que atendesse as necessidades do desenvolvedor. Esta opção encontra-se muito além do objetivo deste projeto, mas sendo uma sugestão para etapas futuras caso seja de interesse a continuação do projeto.
Outra sugestão para continuidade do projeto é a implementação dos testes apresentados para a tecnologia LTE (*Long Term Evolution*). Para este projeto isso não foi possível, pois o aparelho utilizado durante o período de execução do trabalho não era capaz de acessar essa tecnologia.

Um outro caminho para projetos futuros é utilizar algum programa de monitoramento de redes para o caso de utilizar o programa deste projeto para análise de uma região. Isso iria facilitar a tarefa de acompanhar o funcionamento das coletas e dados e podendo gerar de formar automática relatórios ou alertas em caso de queda de qualidade em uma grande quantidade de pontos de determinada região.

# 7 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Exame. CLARO. Tópicos. Disponível em < http://exame.abril.com.br/topicos/claro >. Acesso em: 10 ago. 2014

[2] Exame. OI. Tópicos. Disponível em < http://exame.abril.com.br/topicos/oi >. Acesso em: 10 ago. 2014

[3] Exame. TIM. Tópicos. Disponível em < http://exame.abril.com.br/topicos/tim >. Acesso em: 10 ago. 2014

[4] Exame. VIVO. Tópicos. Disponível em < http://exame.abril.com.br/topicos/vivo >. Acesso em: 10 ago. 2014

[5] Anatel. Participação do Mercado por UF. Disponível em <http://sistemas.anatel.gov.br/SMP/Administracao/Consulta/ParticipacaoMercado/TelaResultado.asp?acao=c&intMes=05&intAno=2014>. Acesso em: 10 ago. 2014

[6] IBGE. Projeção da população do Brasil e das Unidades da Federação. Disponível em <http://www.ibge.gov.br/apps/populacao/projecao/index.html >. Acesso em: 10 ago. 2014

[7] Administradores. As Cinco Forças de Porter. Disponível em <http://www.administradores.com.br/artigos/economia-e-financas/as-cinco-forcas-de-porter/57341/ >. Acesso em: 10 ago. 2014

[8] INVESTOPEDIA. The Industry Handbook: The Telecommunications Industry. Disponível em < http://www.investopedia.com/features/industryhandbook/telecom.asp >. Acesso em: 10 ago. 2014

[9] Anatel. Sobre a Anatel. Disponível em <http://www.anatel.gov.br/Portal/exibirPortalInternet.do >. Acesso em: 10 ago. 2014

[10] Eclipse. Eclipse IDE for Java Developers. Disponível em <https://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/lunar >. Acesso em: 11 ago. 2014

[11] Android. Get the Android SDK. Disponível em <http://developer.android.com/sdk/index.html >. Acesso em: 11 ago. 2014

[12] Ubuntu. Meet Ubuntu. Disponível em < http://www.ubuntu.com/desktop >. Acesso em: 11 ago. 2014

[13] Google play. SSHDroid. Disponível em <https://play.google.com/store/apps/details?id=berserker.android.apps.sshdroid&hl=en >. Acesso em: 11 ago. 2014

[14] Presidência da República – Casa Civil. LEI No 9.472, DE 16 DE JULHO DE 1997. Disponível em < http://www.planalto.gov.br/ccivil_03/leis/l9472.htm >. Acesso em: 11 ago. 2014

[15] Portal Brasil. Agências reguladoras fiscalizam a prestação de serviços públicos. Disponível em < http://www.brasil.gov.br/governo/2009/11/agencias-reguladoras >. Acesso em: 11 ago. 2014

[16] Anatel. Portaria nº 445, de 3 de junho de 2014. Disponível em <http://legislacao.anatel.gov.br/procedimentos-de-fiscalizacao/771-portaria-445 >. Acesso em: 10 ago. 2014

[17] Wikipedia. Secure *Shell*. Disponível em <http://en.wikipedia.org/wiki/Secure_Shell >. Acesso em: 11 ago. 2014

[18] Sourceforge. Non-interactive ssh password auth. Disponível em <http://sourceforge.net/projects/sshpass/>. Acesso em: 11 ago. 2014

[19] Google play. G-Mon. Disponível em <https://play.google.com/store/apps/details?id=de.carknue.gmon2&hl=en>. Acesso em: 11 ago. 2014

[20] Stack Overflow. How to switch from 3G to 2G in android on button clicked. Disponível em <http://stackoverflow.com/questions/24696167/how-to-switch-from-3g-to-2g-in-android-on-button-clicked?rq=1 >. Acesso em: 27 jul. 2014

[21] Stack Overflow. How can I get the dreaded WRITE_SECURE_SETTINGS permission for my android app? Disponível em <http://stackoverflow.com/questions/5034160/how-can-i-get-the-dreaded-write-secure-settings-permission-for-my-android-app>. Acesso em: 27 jul. 2014

[22] Stack Overflow. Want to disable GPS setting through my app. Disponível em <http://stackoverflow.com/questions/9790733/want-to-disable-gps-setting-through-my-app?lq=1>. Acesso em: 27 jul. 2014

# APÊNDICE A - CÓDIGO COMPLETO DO APLICATIVO ANDROID

```
package com.NeighboringCellInformation;

import java.io.File;
import java.util.List;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.io.FileReader;
import java.io.IOException;
import android.widget.Toast;
import java.io.BufferedReader;
import    java.lang.String;
import    java.lang.Integer;

import android.util.Log;
import android.net.Uri;
import android.content.Intent;

import java.util.Timer;
import java.util.TimerTask;

import android.annotation.SuppressLint;
import android.annotation.TargetApi;
import android.app.Activity;
import android.content.Context;
import android.os.Build;
import android.os.Bundle;
import android.telephony.NeighboringCellInfo;
import android.telephony.TelephonyManager;
import android.telephony.PhoneStateListener;
import android.widget.TextView;
import android.telephony.CellInfo;
//import android.os.Environment;


@SuppressLint("NewApi")
@TargetApi(Build.VERSION_CODES.JELLY_BEAN_MR1)
```

```
public class NeighboringCellInformation extends Activity {

    //**************************************************************************
    //********** salva arquivo na memória externa ******************************
    //**************************************************************************
    public void fSaveFile(String myFileText) {
            try {
       File myFile = new File("/sdcard/PG/android_file.txt");
       myFile.createNewFile();
       FileOutputStream fOut = new FileOutputStream(myFile);
       OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);
       myOutWriter.append(myFileText);
       myOutWriter.close();
       fOut.close();

    } catch (Exception e) {
       Toast.makeText(getBaseContext(), e.getMessage(),
            Toast.LENGTH_SHORT).show();
    }

     }

    //**************************************************************************
    //********** lê arquivo da memória externa *********************************
    //**************************************************************************
    public String fReadFile(String myFileText) {

            File sdcard = new File("/sdcard/PG/");

            File file = new File(sdcard,myFileText);

            StringBuilder text = new StringBuilder();

            try {
               BufferedReader br = new BufferedReader(new FileReader(file));
               String line;

               while ((line = br.readLine()) != null) {
                  text.append(line);
                  text.append('\n');
```

```
            }
        }
        catch (IOException e) {

        }

        String r = text.toString();
        return r;

}


//*************************************************************************
//********** lê qual comando para executar ********************************
//*************************************************************************
public String fReadTechPara() {

        String var_r = "";

        String s_techpara = fReadFile("android_read.txt");

   if (s_techpara.contains("WV")) {
        var_r = "WV";
   } else if (s_techpara.contains("WC")) {
        var_r = "WC";
   } else if (s_techpara.contains("WD")) {
        var_r = "WD";
   } else if (s_techpara.contains("GV")) {
        var_r = "GV";
   } else if (s_techpara.contains("GC")) {
        var_r = "GC";
   }

        return var_r;
}

//*************************************************************************
//********** obtém informações de células *********************************
//*************************************************************************
public String fCellInfo (){
```

```
                TelephonyManager telephonyManager =
(TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);
                List<CellInfo> Cinfolist = telephonyManager.getAllCellInfo();

                CellInfo cCell = Cinfolist.get(0);
                String s_cCell = cCell.toString();

                return s_cCell;

        }

        //********************************************************************************
        //********** realiza chamada  ****************************************************
        //********************************************************************************
        public void fMakeCall (){
                PhoneCallListener phoneListener = new PhoneCallListener();
                TelephonyManager telephonyManager =
(TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);
                telephonyManager.listen(phoneListener,PhoneStateListener.LISTEN_CALL_STATE);

                Intent callIntent = new Intent(Intent.ACTION_CALL);
                callIntent.setData(Uri.parse("tel:9xxxxXXXX"));
                startActivity(callIntent);

        }

        //********************************************************************************
        //********** parte central do aplicativo ***************************************
        //********************************************************************************
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_neighboring_cell_information);

     //TextView Neighboring = (TextView)findViewById(R.id.test_1);

     Timer timer = new Timer();

     String s_techpara= fReadTechPara();
```

```
    if (s_techpara == "WC" || s_techpara == "GC"){

            timer.schedule(new TimerTask() {
            @Override
            public void run(){
                    TimerThreadWC();
            }
        }, 1000, 60000);

    }else if (s_techpara == "WD"){

    }else {

            timer.schedule(new TimerTask() {
            @Override
            public void run(){
                    TimerThread();
            }
        }, 0, 2000);
    }
}

//*****************************************************************************
//********** Thread realacionada ao timer ************************************
//********** Não possui permissão para modificar UI **************************
//*****************************************************************************
private void TimerThread(){
        this.runOnUiThread(TimerThread_onUI);
}

private void TimerThreadWC(){
        this.runOnUiThread(TimerThread_onUI);
}
//*****************************************************************************
//********** Método que funciona na mesma thread da UI ***********************
//********** Permitido modificar UI ******************************************
//*****************************************************************************
private Runnable TimerThread_onUI = new Runnable() {
        public void run (){
```

```
TextView Neighboring = (TextView)findViewById(R.id.test_1);
TextView textTitle = (TextView)findViewById(R.id.title);

String s_command = fReadTechPara();

switch(s_command) {
	case "WV":
		String cWcdmaInfo = fCellInfo();

		String s_testTitle1 = "UMTS voice test";
		textTitle.setText(s_testTitle1);

		int index_cid = cWcdmaInfo.indexOf(" mCid=");
		int index_ss = cWcdmaInfo.indexOf("ss=");
		int index_lac = cWcdmaInfo.indexOf("mLac=");
		int index_psc = cWcdmaInfo.indexOf(" mPsc=");
		int index_endss = cWcdmaInfo.indexOf(" ber=");
		int index_endpsc = cWcdmaInfo.indexOf("} C");

		String var_Cid = cWcdmaInfo.substring(index_cid+6, index_psc);
		var_Cid = String.valueOf(Integer.parseInt(var_Cid) - 14876672);
		String var_SS = cWcdmaInfo.substring(index_ss+3, index_endss);
		var_SS = String.valueOf(2*(Integer.parseInt(var_SS))-113);
		String var_Lac = cWcdmaInfo.substring(index_lac+5, index_cid);
		String var_Psc = cWcdmaInfo.substring(index_psc+6, index_endpsc);

		String s_toPc = var_Cid + "\n" + var_SS + "dBm\n" + var_Lac + "\n" +
var_Psc + "\n";
		String s_toApp = "Cell ID: " + var_Cid + "\nSignal Strengh: " + var_SS +
"dBm\nLAC: " + var_Lac + "\nPSC: " + var_Psc + "\n";
		Neighboring.setText(s_toApp);
		fSaveFile(s_toPc);
		break;

	case "WC":
		String s_testTitle2 = "UMTS call test";
		textTitle.setText(s_testTitle2);

		fMakeCall();
```

```
                    break;

                case "GV":
                    String cGsmInfo = fCellInfo();

                    String s_testTitle3 = "GSM voice test";
                    textTitle.setText(s_testTitle3);

                    int index_gcid = cGsmInfo.indexOf(" mCid=");
                    int index_gss = cGsmInfo.indexOf("ss=");
                    int index_glac = cGsmInfo.indexOf(" mLac=");
                    int index_gendcid = cGsmInfo.indexOf("} Cell");
                    int index_gendss = cGsmInfo.indexOf(" ber=");

                    String var_gCid = cGsmInfo.substring(index_gcid+6, index_gendcid);
                    String var_gSS = cGsmInfo.substring(index_gss+3, index_gendss);
                    var_gSS = String.valueOf(2*(Integer.parseInt(var_gSS))-113);
                    String var_gLac = cGsmInfo.substring(index_glac+6, index_gcid);

                    String s_gtoPc = var_gCid + "\n" + var_gSS + "dBm\n" + var_gLac + "\n";

                    String s_gtoApp = "Cell ID: " + var_gCid + "\nSignal Strengh: " + var_gSS
+ "dBm\nLAC: " + var_gLac + "\n";

                    Neighboring.setText(s_gtoApp);
                    fSaveFile(s_gtoPc);

                    break;

                case "GC":
                    String s_testTitle4 = "GSM voice test";
                    textTitle.setText(s_testTitle4);

                    fMakeCall();

                    break;
            }

        }
```

```
  };

//*********************************************************************************
//********** classe para salvar informações de chamada ***********************************
//*********************************************************************************

  private class PhoneCallListener extends PhoneStateListener {

                String LOG_TAG = "LOGGING 123";

                @Override
                public void onCallStateChanged(int state, String incomingNumber) {

                        if (TelephonyManager.CALL_STATE_RINGING == state) {
                                Log.i(LOG_TAG, "RINGING, number: " + incomingNumber);
                                fSaveFile("RINGING\n");
                        }

                        if (TelephonyManager.CALL_STATE_OFFHOOK == state) {
                                Log.i(LOG_TAG, "OFFHOOK");
                                fSaveFile("OFFHOOK\n");
                        }

                        if (TelephonyManager.CALL_STATE_IDLE == state) {
                                // run when class initial and phone call ended,
                                // need detect flag from CALL_STATE_OFFHOOK
                                Log.i(LOG_TAG, "IDLE");
                                fSaveFile("IDLE\n");
                        }
                }
        }
}
```

# APÊNDICE B - CÓDIGO COMPLETO DO PROGRAMA EM *SHELL SCRIPT*

```
#!/bin/bash
clear

pcode="admin"
paddress="192.168.25.12"
pport="2222"

c_seg=1 #constante para aguardar 1 segundo

trap '{ flag_end=0; echo -en "\033[9B"; clear; sshpass -p $pcode ssh -p $pport root@$paddress cp /sdcard/PG/andr_in.txt /sdcard/PG/android_read.txt; }' INT

#****************************
#**** funcoes ****************
#****************************

fCallGV(){
	flag_end=1
	while [ $flag_end == 1 ]
	do
		sshpass -p $pcode ssh -p $pport root@$paddress cat /sdcard/PG/android_file.txt > /home/rafael/PG_U/pc_read.txt
		var_line=1

		while read pc_file
		do
			case $var_line in
			"1") var_CellID=$pc_file ;;
			"2") var_SS=$pc_file ;;
			"3") var_LAC=$pc_file ;;
			# "4") var_PSC=$pc_file ;;
			esac

			((var_line++))
		done < "pc_read.txt"
```

```
		echo -en "\033[sCell Id: $var_CellID   Signal Strenght: $var_SS     \nLAC: $var_LAC \033[u"
		sleep $c_seg

	done

}

fCallWV(){
	flag_end=1
	while [ $flag_end == 1 ]
	do
		sshpass -p $pcode ssh -p $pport root@$paddress cat /sdcard/PG/android_file.txt > /home/rafael/PG_U/pc_read.txt
		var_line=1

		while read pc_file
		do
			case $var_line in
			"1") var_CellID=$pc_file ;;
			"2") var_SS=$pc_file ;;
			"3") var_LAC=$pc_file ;;
			"4") var_PSC=$pc_file ;;
			esac

			((var_line++))
		done < "pc_read.txt"

		echo -en "\033[sCell Id: $var_CellID   Signal Strenght: $var_SS     \nLAC: $var_LAC      PSC: $var_PSC \033[u"
		sleep $c_seg
		#((v_a++))

	done

}

fCallWC(){

	flag_end=1
```

```
	var_cseg=0
	var_calldrop=0
	var_callgood=0

	while [ $flag_end == 1 ]
	do
		sshpass -p $pcode ssh -p $pport root@$paddress cat /sdcard/PG/android_file.txt > /home/rafael/PG_U/pc_read.txt
		while read pc_file
		do
			var_state=$pc_file

		done < "pc_read.txt"

		echo -en "\033[sCall State: $var_state \nDropped Call: $var_calldrop \nSuccessful Call: $var_callgood \033[u"

		if [ $var_state == "OFFHOOK" ]
		then
			((var_cseg++))
		else
			if [ $var_cseg != 0 ]
			then
				if [ $var_cseg -lt 30 ]
				then
					((var_calldrop++))
				else
					((var_callgood++))
				fi
				var_cseg=0
			fi
		fi

		sleep $c_seg
	done
}

#*************************
#**** inicio do codigo *******
#*************************
```

```
#escolher tecnologia
echo -en "Choose one of the options below:\n [1]\tWCDMA voice test\n [2]\tWCDMA call test\n"
echo -en " [3]\tGSM voice test\n [4]\tGSM call test\n\n [0]\texit\n"

sshpass -p $pcode ssh -p $pport root@$paddress cp /sdcard/PG/andr_in.txt /sdcard/PG/android_read.txt

#pegar resposta teclado
var_tech=""
flag_tech=0

while [ $flag_tech == 0 ]
do
	read -n 1 var_tech
	case $var_tech in
		"1") flag_tech=1
			var_tp="WV"
			clear
			echo -en "WCDMA voice test\n\n";;
		"2") flag_tech=1
			var_tp="WC"
			clear
			echo -en "WCDMA call test\n\n";;
		"3") flag_tech=1
			var_tp="GV"
			clear
			echo -en "GSM voice test\n\n";;
		"4") flag_tech=1
			var_tp="GC"
			clear
			echo -en "GSM call test\n\n";;
		"0") clear; exit;;
	esac
	echo -e -n "\033[1D"
done


case $var_tp in
	"GV") sshpass -p $pcode ssh -p $pport root@$paddress cp /sdcard/PG/GV.txt
/sdcard/PG/android_read.txt
```

```
        fCallGV;; #chama funcao GSM voz

        "GC") sshpass -p $pcode ssh -p $pport root@$paddress cp /sdcard/PG/GC.txt
/sdcard/PG/android_read.txt
        fCallWC;; #chama funcao GSM chamada

        "WV") sshpass -p $pcode ssh -p $pport root@$paddress cp /sdcard/PG/WV.txt
/sdcard/PG/android_read.txt
        fCallWV;; #chama funcao WCDMA voz

        "WC") sshpass -p $pcode ssh -p $pport root@$paddress cp /sdcard/PG/WC.txt
/sdcard/PG/android_read.txt
        fCallWC;; #chama funcao WCDMA chamada

esac
```