Implementação Embarcada de Detectores de Eventos Acústicos Aplicados à Televigilância Médica usando Plataformas Móveis Baseadas no Sistema Operacional Android

Allan Brito dos Santos Dantas[1], Guilherme Boroni Pereira[1], Marcos André Aragão[1], Edson Barbosa Lisboa[2], Fábio Luiz Sá Prudente[3]

[1]Alunos do Curso Técnico em Informática do IFS. Bolsistas do PIBIC Jr. e-mail: allan.brito95@gmail.com, guilherme.boroni@hotmail.com, xocomarcos@hotmail.com
[2]Doutor em Engenharia da Computação e professor do IFS. e-mail: edbarbosa@gmail.com
[3]Mestre em Engenharia da Computação e professor do IFS. e-mail: fprudente@gmail.com

Resumo: A televigilância médica trata do monitoramento remoto de pacientes. O uso de sensores simples, como microfones, em monitoramento de ambientes, tem provado ser particularmente útil na detecção de situações de pânico ou dor, principalmente quando o indivíduo sob monitoramento no ambiente sofre de alguma deficiência cognitiva. Nesses cenários de aplicações possíveis, o uso do processamento de sinais sonoros se destaca pela sua simplicidade, através da detecção de sons anormais tais como aqueles que, dentro de apartamentos residenciais, estão associados a situações de acidentes e perigos potenciais, e.g. quedas de objetos, gritos, gemidos de dor, etc. Atualmente, a capacidade de processamento e de memória disponíveis em muitos telefones celulares, principalmente nos chamados smartphones, associadas à padronização para o desenvolvimento de aplicativos trazida pela plataforma Android, fazem com que esses aparelhos sejam ideais também para a função de processamento e reconhecimento de sinais sonoros para fins de televigilância médica. O objetivo desse projeto de iniciação científica foi implementar um aplicativo de reconhecimento de padrões sonoros em um smartphone com o sistema operacional Android, para fins de televigilância médica, e avaliar sua eficácia em diversas condições de ambientes (ruídos, reverberações...).

Palavras–chave: android, detecção de eventos acústicos, televigilância médica, sistemas embarcados

1. TELEVIGILÂNCIA MÉDICA

A televigilância médica trata do monitoramento remoto de pacientes, e a automação desse monitoramento é uma área de pesquisa rica em questões abertas. Dentro desse foco de pesquisa, a detecção e classificação de eventos acústicos tem desempenhado um papel relevante, como uma sub-área recente da computational auditory scene analysis [VALENZISE ET al, 2007], onde uma atenção particular tem sido dada aos sistemas de vigilância automáticos [WANG ET al, 2006], [ZIEGER ET al, 2008], [FEIL-SEIFER ET al, 2005].

Mais especificamente, o uso de sensores simples, como microfones, em monitoramento de ambientes tem provado ser particularmente útil na detecção de situações de pânico ou dor, principalmente quando o indivíduo sob monitoramento no ambiente sofre de alguma deficiência cognitiva. Vale notar que o monitoramento continuado de pacientes pode ter outras aplicações além da detecção de situações de urgência, pois pesquisas recentes em medicina [ROUTROU ET al] concluíram que distúrbios cognitivos leves porém persistentes podem ser indícios prévios do desenvolvimento de doenças como o Mal de Alzheimer. Paralelamente, a detecção eficiente de situações de emergência ou pânico é uma função central em tecnologia de robótica de assistência [ROUAS ET al, 2006], que busca prover assistência cognitiva personalizada.

Em todos esses cenários de aplicações possíveis, o uso do processamento de sinais sonoros se destaca pela sua simplicidade, através da detecção de sons anormais tais como aqueles que, dentro de apartamentos residenciais, estão associados a situações de acidentes e perigos potenciais, e.g. quedas de objetos, gritos, gemidos de dor, etc. O estudo dos sinais sonoros, isolados ou acompanhados do estudo de imagens, permite a obtenção de informações importantes na detecção de situações "anormais". No entanto, as implementações práticas geralmente são acompanhadas de problemas de robustez, em presença de ruídos acústicos captados pelos microfones que servem como sensores acústicos desses sistemas. Assim, do ponto de vista teórico, a segmentação e classificação digital de

ISBN 978-85-62830-10-5

sinais sonoros constituem um problema para o qual as soluções propostas na literatura crescem em termos de criatividade e complexidade à medida em que as capacidades de cálculo e de memória dos processadores digitais crescem.

2. MATERIAL E MÉTODOS

O projeto de pesquisa foi dividido em 4 etapas: estado da arte, desenvolvimento de aplicações teste para Android, desenvolvimento de aplicações de televigilância médica e avaliação dos resultados/conclusão.

No Estado da Arte, foram identificadas as ferramentas e ambientes de desenvolvimento que poderiam ser utilizados para a plataforma, bem como as bibliotecas, funções e APIs (Application Programming Interfaces) do Android SDK(Software Development Kit).

Na segunda etapa, foram implementadas aplicações de teste para avaliar o comportamento e desempenho da plataforma Android. Os testes dos aplicativos desenvolvidos durante o projeto fizeram uso de 02 smartphones Motorola Milestone 2, com sistema operacional Android 2.2.

Ao mesmo passo em que as aplicações teste eram desenvolvidas e tomavam o rumo de processamento sonoro estavam sendo desenvolvidos pelos colaboradores da UFS os algoritmos específicos de reconhecimento de padrões sonoros com a finalidade de monitoramento e televigilância médica.

A etapa final foi concluída com a confecção de relatórios com o objetivo de avaliar os resultados obtidos, e apresentar as conclusões.

3. CARACTERÍSTICAS DOS DISPOSITIVOS MÓVEIS

Um dispositivo móvel compreende qualquer equipamento ou periférico com dimensões e peso que permitem o seu transporte com facilidade e comodidade, baixo consumo de energia e cujas funcionalidades de hardware/software possibilitam o seu uso em diversos contextos operacionais. São exemplos de tais dispositivos: samrtphones, celulares, pagers, PDAs (Assistentes Digitais Pessoais) dentre outros [COSTA, 2012].

Os smartphones nasceram de uma evolução natural das necessidades móveis dos usuários modernos. Com o advento dos PDAs, as pessoas começaram a levar suas informações e compromissos consigo onde quer que fossem, então foi mais do que natural convergir várias funcionalidades em apenas um dispositivo. Surgiu então o conceito de smartphone, um celular que não servia apenas para ligações, mas também como ferramenta de trabalho.

O primeiro smartphone foi desenvolvido pela IBM em 1992 e lançado para o público em 1993. Se chamava Simon e além de ser um celular, continha um calendário, lista de endereços, relógio mundial, calculadora, bloco de notas, e-mail, fax, jogos e já utilizava a tecnologia touchscreen[SMARTPHONECELULAR].

Um smartphone possui uma plataforma, o sistema operacional, sobre a qual programadores são capazes de desenvolver aplicações, permitindo instalar e rodar os mais variados softwares expandindo assim suas funcionalidades. A maioria dos smartphones suportam capacidades de e-mail aliadas a funcionalidade de um personal organizer (organizador pessoal)., possuem teclado QWERTY (teclado alfanumérico) e/ou tecnologia touchscreen, gerenciador de contatos, sistema operacional próprio, habilidade de visualizar qualquer tipo de documento (PDF, Microsoft Office, etc.), browser para navegação na internet, funções multimídia para executar arquivos de música, vídeo e fotos[SMARTPHONECELULAR].

A evolução digital e a integração em larga escala têm dotado esses dispositivos com um poder computacional equivalente ou superior a um computador pessoal (desktop) atual com uma robusta configuração. Uma configuração típica de tais dispositivos compreende um processador com poder computacional igual ou superior a 600 MIPS (Milhões de Instruções por Segundo), por exemplo ARM11, memória RAM, memórias não voláteis (flash), expansão de memória (micro SD), interfaces de comunicação (USB, WIFI, 3g), displays sensíveis ao toque (touch screen) com diferentes dimensões, microfones para captura de sinais e alto falantes. Desta forma, a grande maioria dos dispositivos móveis atuais se credenciam para executar aplicações que exijam um alto poder

ISBN 978-85-62830-10-5

computacional, tais como processamento de sinais[VELDHUIZEN, 2012], foco desse trabalho. Todas essas características, aliadas ao custo relativamente baixo dos dispositivos móveis, tornam esses equipamentos uma solução atrativa para diversas aplicações que exijam uma solução integrada de hardware e software (sistemas embarcados), como é o caso da televigilância médica.

O uso de um smartphone para esse fim é vantajoso também, por ser um aparelho de uso cada vez mais comum, que naturalmente é mantido próximo ao usuário, e por ser ele próprio capaz de efetuar uma ligação telefônica, ou enviar uma mensagem de texto, caso detecte algum padrão sonoro considerado anormal ou indicativo de situação de emergência, a fim de chamar socorro imediato.

4. ARQUITETURA DO SISTEMA OPERACIONAL ANDROID

O Android é um conjunto completo de softwares para dispositivos móveis que inclui sistema operacional e importantes aplicativos. É um projeto de código aberto e foi idealizado para aparelhos com especificações diversas [OHA, 2010]. Surgiu sob a forma de plataforma open source como uma resposta da Google à crescente demanda de recursos no mundo tecnológico móvel, introduzindo os conceitos de integração e flexibilidade. [PEREIRA et. all, 2009].

Um dos fatores que contribui para a disseminação do sistema é o fato de que a Google vem lançando concursos no desenvolvimento de aplicativos para a plataforma, que oferecem prêmios atrativos para os desenvolvedores do mundo inteiro. No maior concurso da área o Android Developer Challenger, o investimento foi de U$ 10 milhões em prêmios, distribuindo uma quantia de U$ 275.000,00 para as 20 melhores aplicações. O grande impacto para a comunidade de usuários do sistema operacional, é a quantidade crescente de aplicativos construídos durante o processo de avaliação, e de que a maioria destes códigos são marcados como livre por seus criadores, estando ao acesso de uma gama de usuários. O Android conta com uma loja virtual, a Google Playstore, que possibilita ao desenvolvedor disponibilizar uma aplicação gratuitamente para o usuário final, ou lucrar com a sua venda. [ARIMA, 2009]. Outras características também são relevantes como a fácil integração com serviços oferecidos pela Google. Além disso, diversas ferramentas estão disponíveis gratuitamente para os seus desenvolvedores [MEIER, 2009]. Atualmente a plataforma Android corresponde aproximadamente 68,1% do mercado de soluções embarcadas, tornando um mercado muito atrativo para as aplicações e soluções baseadas nessa plataforma[IG SÃO PAULO, 2012].

Android em si é uma pilha de software (Software Stack), ou "arquitetura de pilha de software", laminado dividido em três camadas: sistema operacional, middleware e aplicações. A arquitetura do sistema é divida nos seguintes componentes: aplicativos, framework nativo, bibliotecas, runtime android, kernel linux[Figura 1][OFICINA DA NET, 2010].

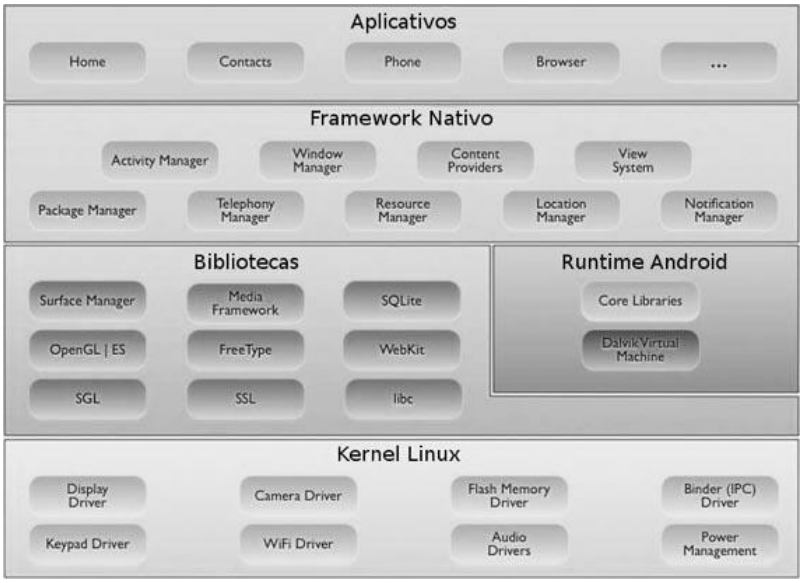

Figura 1 – Arquitetura do sistema Android

ISBN 978-85-62830-10-5

Aplicativos. A camada de aplicativos é a camada mais alta da arquitetura do sistema, composta pelo conjunto de aplicativos nativos do sistema, dentre estes pode se citar: cliente de e-mail, calendário, mapas, browser e internet, despertador, jogos, e outros.

Framework Nativo. A camada de framework nativo disponibiliza aos desenvolvedores as mesmas APIs (Application Programming Interface) usadas para criar as aplicações originais do sistema, não necessitando ao programador conhecimento detalhado de sua implementação, apenas usar seus serviços [DICIONÁRIO WEB, 2009].

Este framework foi criado para abstrair a complexidade e simplificar o reuso de procedimentos. Essa camada funciona como um meio de ligação com a camada de bibliotecas do sistema que serão acessadas através de APIs contidas no framework.

Entre as APIs do Android podemos destacar: Location Manager: usada para obter a posição geográfica do usuário; Telephony Manager: informações sobre dispositivos como bateria e serviços de telefonia celular podem ser obtidos através dessa API; Window Manager: responsável pelo gerenciamento de toda janela de uma aplicação; Content Providers: responsável pela disponibilização dos dados através das aplicações tornando esses dados públicos; Resource Manager: todos os recursos que uma aplicação irá usar como áudio, vídeo, arquivos XML; Notification Manager: permite que uma aplicação exiba notificações, ative LEDs, luzes, sons ou vibração disponíveis no dispositivo; Mas a API mais importante é a Activity Manager: responsável pelo gerenciamento de cada atividade do sistema. No Android cada atividade é gerenciada através de uma pilha de atividades. Toda nova atividade criada vai para o topo de pilha de atividades e se torna uma running activity, o que quer dizer que será executada. [OFICINA DA NET, 2010]

Bibliotecas. A plataforma também inclui uma coleção de bibliotecas C/C++ usadas pelos componentes do sistema. Essas bibliotecas são acessadas pelo programador via Java, através do framework do sistema. As bibliotecas do núcleo principal são: Sistema biblioteca C, um derivado do BSD, biblioteca padrão do sistema C (libc); Bibliotecas Media, baseado na biblioteca da PacketVideo OpenCORE multimedia; Gerente de superfície, este campo permite que os desenvolvedores acessem o sistema de exibição, e conseguir a integração perfeita entre 2D/3D em múltiplas aplicações; LibWebCore, uma engine avançada de browser web; SGL, motor gráfico 2D; Bibliotecas 3D, um padrão baseado em OpenGL ES 1,0 de engine de gráficos 3D, é possível selecionar a aceleração 3D de hardware ou software altamente otimizado programas 3D para mostrar resultados; SQLite, SQLite é um conjunto de banco de dados relacional open source, que se caracteriza por peso leve (500KB apenas ou mais de capacidade) para o uso de dispositivos portáteis

O Android Runtime permite que cada processo rode sua própria instância da maquina virtual. Contém dois núcleos: Bibliotecas centrais, dentro da biblioteca do núcleo já contém a maior parte da linguagem de programação Java utilizado para efetuar a chamada de função requerida, sendo assim, embora no desenvolvimento de aplicativos seja utilizada a linguagem Java, as aplicações não são executadas em uma maquina virtual Java tradicional, e sim em outra chama de Dalvik. Essa maquina virtual é otimizada especialmente para dispositivos moveis. Dalvik Virtual Machine, não existe uma JVM (Java Virtual Machine) inclusa no sistema, pois o Android tem sua própria maquina virtual, que é otimizada para execução de aplicativos móveis. Essa maquina virtual foi construída pelos engenheiros da Google, para obter um consumo mínimo de memória e isolamento de processos. Ela permite que as aplicações escritas em linguagem Java, sejam executadas normalmente utilizando o mínimo de recursos para implementar e também realizando individuais VM múltiplos.

Kernel Linux. A camada do Kernel é baseada em um Kernel Linux versão 2.6. Esta camada também atua como responsável pela abstração entre o hardware e os aplicativos e responsável pelos principais serviços do sistema como gerenciamento de memória e de processos. Várias funções do kernel são utilizadas diretamente pelo Android tais como: segurança (Security), gerenciamento de memória (Memory Management), Gerenciamento de Processos (Travel Management), Stack de rede, conhecido também como: pilha de rede), Driver Model, etc. Mas muitas modificações foram feitas para otimizar memória e tempo de processamento das aplicações. Essas alterações incluem device drivers novos, adições no sistema de gerenciamento de energia e um sistema que permite finalizar processos de maneira criteriosa quando há pouca memória disponível. O Kernel Linux 2.6 foi

ISBN 978-85-62830-10-5

escolhido por já conter uma grande quantidade de device drivers sólidos e por ter um bom gerenciamento de memória e processos[OFICINA DA NET, 2010].

5. RESULTADOS E DISCUSSÃO

Activity é o componente chave do Android e para haver interação com a interface de usuário, uma classe deve herdar de uma activity. Uma Activity está relacionada à tarefa que uma aplicação pode fazer. Essa tarefa pode ser, por exemplo, uma tela de boas-vindas, um mapa, uma lista de itens, uma tela de opções, algo que possa ser apresentável para o usuário [OFICINA DA NET, 2010] [CELEIRO ANDROID, 2011].

Por meio do método startActivity() pode-se enviar ao Sistema Operacional Android determinados comandos referentes às intenções de realizar alguma ação, os Intents.

Três dos principais componentes de uma aplicação (atividades, serviços e broadcast receivers) são ativados através de mensagens, chamadas intents. Um objeto Intent é uma estrutura de dados passiva que comporta uma descrição abstrata da operação a ser realizada. Ele contém informações de interesse para o componente que o recebe (como uma ação a ser tomada e os dados de argumento) mais informações de interesse para o sistema Android (como qual categoria de componente deve manusear o intent e instruções em como lançar a atividade alvo)[EVERY, 2010][CELEIRO ANDROID, 2011].

O método startActivity(Intent intent) envia dados e requisições de telas. Já o método getIntent() recupera uma intent enviada por meio do startActivity().

A biblioteca Media oferece três formas de gravação de áudio [EVERY, 2010]. A maneira mais fácil, porém menos flexível, é utilizando o método Intent RECORD_SOUND_ACTION da classe MediaStore.Audio.Media[figura 2]. Após a implementação, notou-se que ele permite a gravação e a medição da intensidade do áudio em tempo real, porém é necessário que este permaneça aberto durante todo o processo ocupando a tela. Isso acaba gerando limitações no aplicativo e pode ocasionar ainda na perda de desempenho da aplicação.

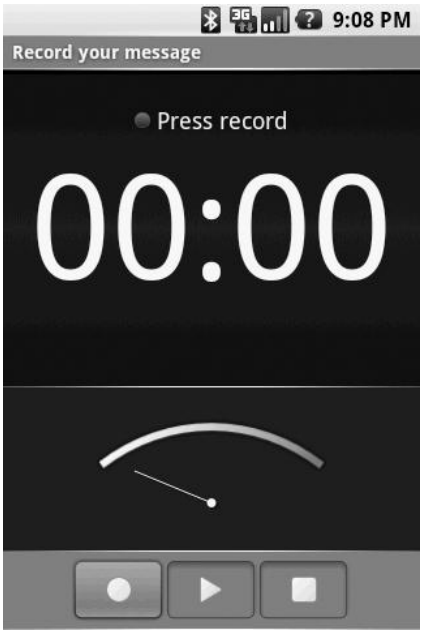

Figura 2 – Aplicativo Intent(MediaStore.Audio.Media.RECORD_SOUND_ACTION)

Outra maneira, um pouco mais flexível, é utilizando a classe MediaRecorder que inclui suporte para codificação e decodificação de uma variedade de tipos de mídias comuns. A classe MediaRecorder possui uma variedade de métodos. O método getMaxAmplitude permite-nos solicitar a amplitude máxima do áudio que foi gravado registrada entre um período de tempo. O valor é reposto

ISBN 978-85-62830-10-5

cada vez que o método é chamado e um medidor de nível de áudio pode ser implementado para chamar esse método periodicamente [EVERY, 2010].

Ao ser implementada, obteve-se como resultado a gravação do áudio do microfone do smartphone em um arquivo no cartão SD, não apenas com a Activity aberta como também rodando em background. Para este último é necessário que a Activity faça uso de uma thread do sistema.

Existem no Android 3 tipos de Threads: o padrão de thread Java e as classe Handler e AsyncTask, específicas do Android. Para o aplicativo de televigilância, a mais indicada é a AsyncTask pois permite executar tarefas de longa duração sem ocupar a interface do usuário e que uma aplicação nao precise de resposta do usuário[VOGEL, 2012].

Foi então criada a classe RecordAmplitude que estende-se na thread AsyncTask,

```
private class RecordAmplitude extends AsyncTask<Void, Integer, Void> {
```

O método doInBackground é executado em um segmento separado e é executado quando o método é chamado no objeto. Este método entra em loop enquanto a gravação está acontecendo e chama o método Thread.sleep(tempo), que faz com que ele nao faça nada por um determinado tempo. Uma vez que está completo, ele chama publishProgress e passa o resultado do método getMaxAmplitude no objeto MediaRecorder.

```
@Override
    protected Void doInBackground(Void... params) {
      while (isRecording) {
        try {
          Thread.sleep(500);
        } catch (InterruptedException e) {e.printStackTrace();
      }
       publishProgress(recorder.getMaxAmplitude());
    }
    return null;
}
```

A chamada anterior ao publishProgress chama o método onProgressUpdate definido aqui, que é executado no thread principal para que ele possa interagir com a interface do usuário. Neste caso, fazemos a atualizaçao de um TextView, chamado de amplitudeTextView, com o valor que é passado a partir da chamada do método publishProgress.

```
protected void onProgressUpdate(Integer... progress) {
   amplitudeTextView.setText(progress[0].toString());
}
```

Após os resultados dos aplicativos com a classe MediaRecorder, avaliou-se que a classe satisfaz a necessidade porém não é tão eficaz quanto poderia pois, ao ser chamada duas vezes, retorna o valor máximo ou mínimo da intensidade entre o intervalo de tempo ao invés de retornar o valor da intensidade do exato momento em que foi chamada. Possivelmente perdendo alguma qualidade no áudio armazenado em buffer. Devido a esta limitação ela foi posta de lado e as pesquisas seguiram para o uso da classe AudioRecord.

A classe AudioRecord é a mais flexível das três classes que nos permite acesso ao fluxo de áudio-primo. Possuindo um menor numero de recursos internos, permite a programação e personalização inclusive de funções básicas, como a que cria o buffer de dados, por exemplo.[APRESS, 2012]

Ela permite a criação direta de um buffer de dados através do método read, evitando assim a possível perda de qualidade no áudio.

```
while(isRecording){
  read = recorder.read(data, 0, bufferSize);
}
```

ISBN 978-85-62830-10-5

Após implementação, avaliou-se que esta é a classe mais indicada para ser usada pelo aplicativo de televigilância médica.

Os aplicativos feitos para Android precisam implementar também permissões ao usuário(O celular). Estas implementações podem ser inseridas através da aba “Permissions” do IDE ou podem também ser inseridas diretamente no arquivo AndroidManifest.xml pela forma de programação com os seguintes comandos:

<uses-permission android:name="android.permission.RECORD_AUDIO"/>

<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

A etapa seguinte será a implementação dos algoritmos específicos de reconhecimento de padrões sonoros que estava, e está ainda, sendo desenvolvida na linguagem MATLAB pelos colaboradores da UFS em paralelo ao projeto.

4. CONCLUSÕES

A primeira etapa desse projeto de pesquisa obteve resultados satisfatórios, propiciando um profundo conhecimento sobre a arquitetura Android.

A continuação desse projeto deverá dar-se explorando a classe Audiorecord em conjunto com os algoritmos específicos de reconhecimentos de padrões sonoros com a finalidade de monitoramento e televigilância médica que estão sendo desenvolvidos pelos colaboradores da UFS.

Os pesquisadores da UFS estão ainda implementando os algoritmos na linguagem MATLAB e testando-os na linguagem C.

A nosso ver, o projeto cumpriu o objetivo de estimular, dentro da comunidade científica e tecnológica do estado de Sergipe, o estudo e o desenvolvimento da televigilância médica. Dando o primeiro passo em pesquisas dessa área no estado e colaborando para que, junto com os colaboradoes da UFS, o objetivo final do projeto seja alcançado.

REFERÊNCIAS

HIROTA, E.H. Desenvolvimento de competências para a introdução de inovações gerenciais na construção através da aprendizagem na ação. 2001. 205p. Tese (Doutorado em Engenharia) - Escola de Engenharia, Universidade Federal do Rio Grande.

VALENZISE, G.; GEROSA, L.; TAGLIASACCHI, M.; ANTONACCI, F.; SARTI, A. Advanced Video and Signal Based Surveillance, in AVSS 2007, vol. 2, issue 5-7, 2007, pp 21-26.

WANG, D.; BROWN, G. Computational Auditory Scene Analysis: Principles, Algorithms and Application, Wiley-IEEE Press, 2006.

ZIEGER, C.; OMOLOGO, M. Acoustic event classification using a distributed microphone network with a GMM/SVM combined algorithm, in Interspeech, set. 2008, pp 115-118.

FEIL-SEIFER, D.; MATARIC, M.J. “Defining socially assistive robotics”, in Proc. IEEE International Conference on Reha-bilitation Robotics (ICORR’05), Chicago, Il, USA, June 2005, pp 465-468.

ROTROU, J.; WENISCH, E.; CHAUSSON, C.; DRAY, F.; FAUCOUNAU, V.; RIGAUD, A.S. Accidental MCI in healthy subjects: a prospective longitudinal study. Eur J Neurol; 12(11):879-85.

ROUAS, J.L.; LOURADOUR, J.; AMBELLOUIS, S. Audio Events Detection in Public Transport Vehicle, in Proc. of the 9th International, IEEEConference on Intelligent Transportation System, Sept. 2006, pp 733-738.

COSTA, C. M., ET al. Monitoramento de plataformas de poços de petróleo através de dispositivos móveis, in HOLOS, ano 28, vol.3, 2012.

ISBN 978-85-62830-10-5

SMARTPHONECELULAR. O que é um smartphone? Disponível em: <http://www.smartphonecelular.com.br/o-que-e-smartphone>. Acesso em: 27 jan. 2012.

VELDHUIZEN, E. J. V., O Android como plataforma de desenvolvimento, in Revista Elektor, mar. 2012, edição brasileira No. 120, pag: 10-12.

OHA. Google Android: aprenda a criar aplicações para dispositivos móveis com o Android SDK. São Paulo: Novatec, 2009. OHA. Alliance Overview. Disponível em: <http://www.openhandsetalliance.com/oha_overview.html>. Acesso em: 26 jun. 2011.

PEREIRA, L. C. O.; SILVA, M. L. Android para Desenvolvedores. Brasport, 2009 - 240 páginas.

ARIMA, K. Android ajuda mercado de smartphones a bater recorde no terceiro trimestre. IDG Now, 6 nov. 2009. Disponível em: <http://idgnow.uol.com.br>. Acesso em: 23 jun. 2011.

MEIER, R.; WERNECK, R. "Professional Android Application Development", in Indianapolis: Wiley Publishing, 2009.

IG SÃO PAULO, iPhone e Android já representam 85% do mercado de smartphones, diz IDC, ago, 2012; Disponível em: <http://tecnologia.ig.com.br/2012-08-08/iphone-e-android-ja-representam-85-do-mercado-de-smartphones-diz-idc.html>. Acesso em: 09 agosto. 2012.

OFICINA DA NET. Google Android OS, ago. 2011. Disponível em: <http://www.oficinadanet.com.br/artigo/outros_sistemas/google_android_os/>. Acesso em: 01 jul. 2011.

DICIONARIO WEB. Disponível em: < http://www.dicionarioweb.com.br/api.html > Acesso em: 27 mai. 2011.

CELEIRO ANDROID. Intents e Intent Filters, abr. 2011. Disponível em: <http://celeiroandroid.blogspot.com/2011/04/intens-e-intent-filters-parte-01.html>. Acesso em: 14 dez. 2011.

EVERY, S. V. E-book. Pro Android Media: Developing Graphics, Music, Video, and Rich Media Apps for Smartphones and Tablets. APRESS. 2012.

VOGEL, L. Android Threads, Handlers and AsyncTask, jan. 2012. Disponível em: <http://www.vogella.de/articles/AndroidPerformance/article.html>. Acesso em: 26 jan. 2012.

ISBN 978-85-62830-10-5