COBENGE 2014
ENGENHARIA: Múltiplos saberes e atuações
16 a 19 de setembro | Juiz de Fora - MG

DESENVOLVIMENTO DE UMA APLICAÇÃO JAVA E ANDROID PARA SIMULAR O MONITORAMENTO DE TRÂNSITO EM SISTEMAS SMART GRID

Leonardo Fernandes Cherubini – cherubini18@hotmail.com
Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso
Departamento Acadêmico de Informática
Rua Professora Zulmira Canavarros, 93
CEP: 78005-200 – Cuiabá – Mato Grosso

Érika Ferreira Moraes Crispim de Menezes – erikafmcm@gmail.com

Valtemir Emerêncio do Nascimento– valtemir.nascimento@cba.ifmt.edu.br

Ruy de Oliveira – ruy.oliveira@cba.ifmt.edu.br

Ed'Wilson Tavares Ferreira – ed@cba.ifmt.edu.br

Resumo: *O presente trabalho mostra o desenvolvimento de uma aplicação Java Desktop capaz de se comunicar com um smartphone Android, utilizando a tecnologia Bluetooth como protocolo de comunicação. A aplicação apresenta um sistema de comunicação piconet, tendo a aplicação Java como master e aplicação Android como slave. Em problemas relacionados a um ecossistema Smart grid, este trabalho apresenta uma solução para melhorar o monitoramento de veículos elétricos em concessionárias de abastecimento de energia, o que torna possível um maior controle da demanda utilizada nas redes elétricas. Em linhas gerais, o aplicativo Java vai funcionar como um ponto de monitoramento Bluetooth e o aplicativo Android como um veículo elétrico. Quando o dispositivo Android passar por um determinado ponto de monitoramento o mesmo enviará um arquivo do tipo txt para o aplicativo master contendo sua velocidade atual. Com os dados obtidos neste trabalho é possível aderir seu uso em salas de aula decorrente ao ecossistema da plataforma Android que passa a trabalhar com conceitos amplamente utilizados na engenharia da computação, como estudos de física, redes e programação.*

Palavras-chave: *Android, Bluetooth, Java, Smart Grid*

1. INTRODUÇÃO

Smart Grid é um termo usado para descrever como elementos de rede elétrica interagem com uma infraestrutura de informação. Uma rede elétrica inteligente gerencia toda a demanda de energia trabalhando em conjunto com sistemas de comunicação. Segundo (Lopes et al,

2012), esta medição inteligente consiste na capacidade dos dispositivos trocarem informações entre si, o que torna a criação de uma rede mais segura e eficiente.

Este trabalho tem como objetivo apresentar uma proposta de como melhorar uma infraestrutura elétrica de monitoramento e recarda de veículos elétricos. As características e limitações sobre veículos elétricos são tratados em maiores detalhes em (Conti et al, 2011).

A tecnologia Bluetooth é a solução proposta neste artigo para retratar uma troca de dados entre um ponto de recarga com o veículo junto ao seu motorista. Em linhas gerais, um experimento utilizando uma rede Bluetooth é apresentado com o objetivo de fomentar a utilização desta tecnologia em situações reais, na qual veículos elétricos são monitorados por pontos de monitoramento Bluetooth localizados em pontos estratégicos em uma determinada região de uma cidade. A intenção deste monitoramento é estudar o comportamento da comunicação entre o dispositivo móvel e a estrutura física. Dessa forma, uma estrutura de monitoramento de veículos terá informações relacionadas a quantidade e a hora em que os veículos são recarregados. Com base nessas informações, uma concessionária de energia conseguirá prever quando e quanto de energia será necessária disponibilizada para a rede elétrica.

Para simular a ideia proposta serão utilizados um notebook que funcionará como o ponto de coleta de dados e um celular smartphone com o sistema operacional Android que fará o papel do carro. Através desta abordagem, pretende-se disseminar entre os alunos a cultura do desenvolvimento de aplicativos que sejam aplicáveis e úteis a sociedade. Portanto, através da abordagem proposta, o objetivo maior é aumentar o interesse e engajamento dos alunos no desenvolvimento de aplicações tecnológicas.

A plataforma Android foi escolhida para este trabalho por ser um sistema gratuito, adaptável para diferentes tipos de dispositivos móveis e por ser muito difundido em ambientes acadêmicos. Já existem propostas para a utilização da plataforma Android em automóveis reais, como pode ser visto em (Wall Street Journal, 2014). A plataforma Java foi escolhida para este projeto por ser uma tecnologia gratuita, pelo tempo de experiência que vem sendo utilizada em ambientes acadêmicos, por ser portável e funcionar em diferentes tipos de *hardware*, podendo ser este um ponto de monitoramento de veículos.

2. MATERIAIS E MÉTODOS

2.1. Fundamentos da plataforma Java e da API Bluecove

A tecnologia Java consiste em uma plataforma de desenvolvimento criada pela empresa Sun Microsystem. Seu lançamento oficial ocorreu no ano de 1995 e a principal proposta desta plataforma foi o mercado de dispositivos eletrônicos. A proposta apresentada pela plataforma Java estava muito distante da realidade dos aparelhos eletrônicos da época e por este motivo o Java não foi difundido neste mercado e sim no mercado de ambientes web (Oracle Corporation, 2014). A plataforma Java tornou-se um padrão mundial para o desenvolvimento de sistemas web, aplicativos de celular, jogos e *softwares* corporativos, utilizando, para isso, uma linguagem de programação orientada a objetos, uma extensa biblioteca e um ambiente de execução seguro e portável.

O Java foi escolhido para este presente trabalho pelo motivo dele ser uma tecnologia multiplataforma, que torna possível a sua utilização em hardwares mais específicos com pouca capacidade de processamento, não se limitando apenas a sistemas operacionais de micro computadores. A aplicação Java desenvolvida neste artigo simula o servidor de um sistema de

monitoramento de transito Smart grid. Em linha gerais, este trabalho apresenta uma aplicação Java desktop desenvolvida a partir das bibliotecas *JavaFx* (*Oracle JavaFX*, 2014) e *Bluecove* (Bluecove, 2014) com o objetivo de trabalhar com a aquisição de dados de uma aplicação cliente via protocolo Bluetooth. A aplicação é ilustrada na figura 1.

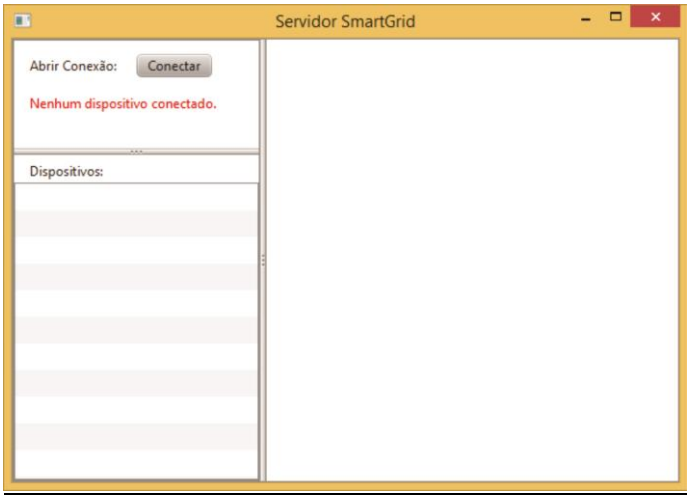

Figura 1: Aplicação do servidor

A biblioteca *JavaFx* consiste na interface gráfica da aplicação com o simples objetivo de exemplificar como funcionará um servidor Smart Grid em execução. A biblioteca *Bluecove*, como é exemplificada no tópico 2.1, é responsável por implementar o protocolo Bluetooth em uma aplicação Java e sendo também responsável pelas principais funcionalidades da aplicação.

2.1.1. Bluecove

A utilização da biblioteca *Bluecove* se deu pelo motivo da mesma adotar à JSR 82. As JSRs (*Java Specification Request*) são especificações que estabelecem como devem ser os recursos atuais e como serão os recursos futuros da plataforma Java. E a JSR 82 representa as especificações voltado para o protocolo *Bluetooth*.

A biblioteca *Bluecove* trata-se de uma API desenvolvida para aplicações Java de código fonte aberto que implementa as principais funcionalidades para a comunicação com dispositivos Bluetooth. Na camada mais alta da estrutura *Bluecove* estão as bibliotecas Bluetooth e CLDC. Estas bibliotecas se interagem com o sistema operacional específico usando a Interface Java JNI. É no sistema operacional que que estão presentes as funcionalidades de comunicação de um dispositivo Bluetooth como a OBEX e a RFCOMM. E é a partir do *Host Controller Interface* (HCI) que o sistema operacional se comunica com o Rádio Bluetooth.

Para a aplicação deste presente trabalho foram utilizadas as classes da API *Bluecove*:

- UUID: responsável por consistir na chave de criptografia UUID;
- *StreamConnectionNotifier*: esta classe cria uma conexão utilizando a chave UUID;
- *Connector*: a classe *Connector* é responsável por abrir uma conexão com uma aplicação cliente;
- *StreamConnection*: esta classe consiste na conexão do servidor com o cliente;
- *RemoteDevice*: consiste no cliente da conexão.
-

Com base na API *Bluecove*, a presente aplicação será capaz de executar as funcionalidades necessárias para o funcionamento do servidor no sistema de monitoramento Smart grid, sendo elas, a procura por dispositivos, a procura pelos serviços requisitados pelo cliente e a realização de aquisição dos dados realizados a partir do dispositivo cliente, tendo todas elas mostradas no tópico 4.0.

2.2. Fundamentos do sistema operacional Android

O Android é um sistema operacional desenvolvido pela Google com o objetivo de funcionar em diferentes plataformas móveis, como celulares, televisores, relógios, geladeiras e até mesmo em alguns modelos de microcontroladores. As principais características da plataforma Android consiste no seu sistema operacional gratuito e no código fonte aberto para a comunidade de desenvolvedores. A plataforma Android é baseada no *kernel* 2.6 do Linux, sendo este responsável pelo gerenciamento de threads, memória e na segurança do sistema.

A arquitetura da plataforma Android é composta por diferentes camadas, isto é: *Applications*, *Application Framework*, *Libraries*, *Android Runtime* e o *kernel* do Linux, conforme representado na figura 2:

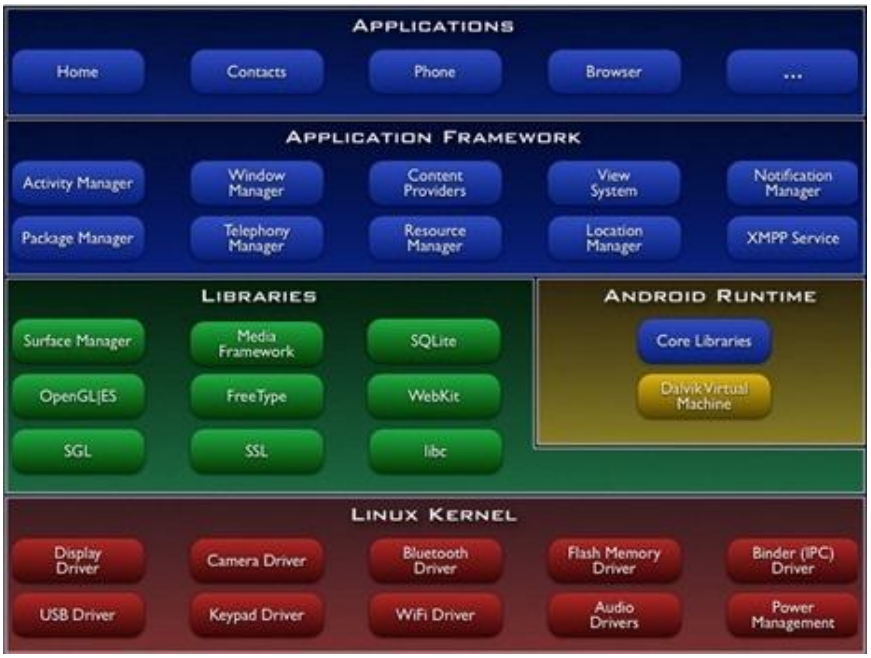

Figura 2: Arquitetura da plataforma Android (ELinux, 2014).

Onde:

- A camada *Applications* são compostas por aplicações nativas do Sistema Android que constituem em aplicativos de calendário, navegador de internet, e-mails, programa SMS como muitos outros;
- A camada Application Framework consiste em componentes da plataforma utilizados para a execução de novas aplicações. São estes, gerenciadores de serviços e de notificações;

- A camada *Libraries* correspondem nas bibliotecas nativas do sistema operacional Android. Entre elas estão as bibliotecas de banco de dados, OpenGL, acelerômetro, fontes *bitmaps* e multimídia;
- A camada Android *Runtime* consiste nos dois componentes necessários para a execução dos aplicativos. O primeiro é o *Core Libraries* que fornece a API Java para o desenvolvimento das aplicações. O segundo é a máquina virtual Dalvik responsável pela execução das aplicações Android, e sendo esta exemplificada com mais detalhes no tópico 3.1;
- Por fim, o *kernel* Linux 2.6 fornece os serviços do núcleo da plataforma Android.

As aplicações do sistema Android são desenvolvidas com o uso da linguagem de programação Java. Mas diferente das aplicações Java convencionais que executa na máquina virtual Java, uma aplicação Android roda em uma máquina virtual Dalvik. O desenvolvimento de uma aplicação Android consiste na criação de códigos fonte Java entre outros arquivos, como XML e diferentes formatos de imagem. Logo após criação de um projeto Android com os arquivos citados anteriormente, o código fonte Java é compilado para um arquivo *bytecode*. Depois do arquivo Java ser compilado, todos os arquivos do projeto são convertidos para o formato Dalvik Executable (extensão .dex), podendo assim ser executado pela máquina virtual Dalvik, conforme ilustra a figura 3.

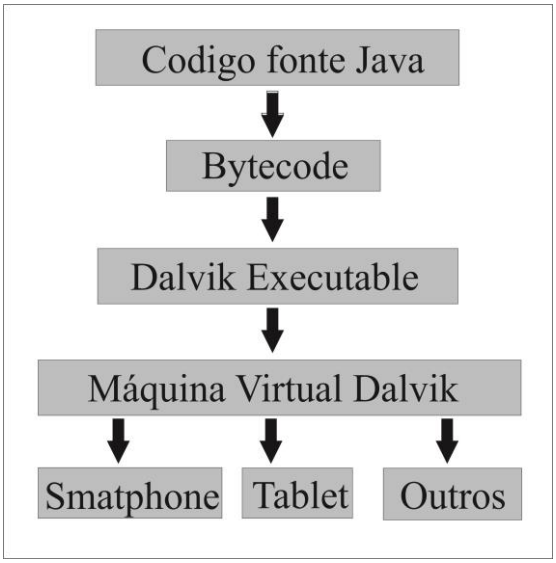

Figura 3: Execução de uma aplicação Android.

A máquina virtual Dalvik foi desenvolvida pela Google e tem como seu principal autor Dan Bornstein. Dalvik foi projetada exclusivamente para o sistema Android e é otimizada para execução em plataformas móveis.

Esta máquina virtual é otimizada para utilizar menos memória e é projetada para que diversas instâncias de máquinas virtuais executem ao mesmo tempo de forma eficiente.

A aplicação proposta para este trabalho tem como objetivo simular a operação cliente em um sistema de monitoramento de trânsito *Smart grid*. Uma aplicação do sistema operacional Android consistirá no papel de um automóvel no sistema *Smart grid*. Um smartphone será utilizado para a captura de informações do respectivo automóvel simulado na aplicação

Android, e para a execução deste aplicativo serão utilizadas as bibliotecas Bluetooth (Android Bluetooth, 2014) e de Acelerômetro (Android Motion Sensors, 2014) do Android.

2.2.1. Biblioteca Bluetooth

A plataforma Android possui suporte a rede Bluetooth, permitindo que um dispositivo troque dados com outros dispositivos via protocolo Bluetooth. A estrutura de uma aplicação Android tem acesso as funcionalidades Bluetooth por meio das APIs Android. As principais funcionalidades fornecidas pela API de Bluetooth são:

- Procurar por outros dispositivos Bluetooth;
- Consultar os dispositivos emparelhados através do adaptador Bluetooth;
- Iniciar canais RFCOMM;
- Conexão com outros dispositivos por meio do serviço de descoberta;
- Transferência de dados om outros dispositivos;
- Gerenciamento de múltiplas conexões.

As classes da biblioteca Bluetooth utilizadas para o desenvolvimento da aplicação Android são apresentadas a seguir:

- *BluetoothAdaptor*: Consiste em uma instância do rádio Bluetooth de dispositivo móvel;
- *BluetoothDevice*: Define uma instância do dispositivo conectado a aplicação;
- *BluetoothSocket*: É utilizado como ponto de conexão para a troca de dados com outro dispositivo.

Tomando como base as classes citadas anteriormente, o aplicativo será capaz de reconhecer uma conexão com uma aplicação servidor, conectar-se a aplicação e enviar os dados necessários para o funcionamento do sistema *Smart grid*.

2.2.2. Biblioteca Sensor de Movimento

A plataforma Android fornece diversos sensores que permite o controle do movimento de um dispositivo móvel. Os sensores de movimento são importantes para o monitoramento da inclinação, vibração e rotação do dispositivo. Todos os sensores constituem em matrizes multidimensionais. Por exemplo, o sensor acelerômetro retorna valores da aceleração dos três eixos de uma determinada coordenada para dentro de uma matriz.

Para este trabalho é utilizado apenas o sensor de aceleração, sendo este suficiente para o cálculo da velocidade do dispositivo. O sensor acelerômetro (Fg) dividida pela massa (*mass*) mede a aceleração aplicada no dispositivo (Ad), como pode ser vista na equação (1):

$$A_d = -\Sigma F_g / \text{mass} \quad (1)$$

No entanto, a força da gravidade pode influenciar a medida da aceleração, como pode ser visto na equação (2):

$$A_d = -g - \Sigma F_g / \text{mass} \quad (2)$$

Para o cálculo da velocidade do dispositivo móvel será utilizado apenas a multiplicação da aceleração pelo tempo. Na figura 4 pode ser visto como os eixos de

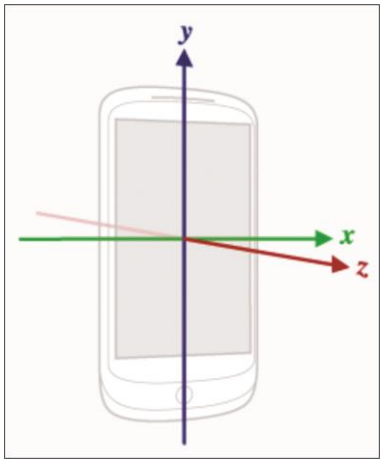

Figura 4: Eixos x,y e z em relação ao dispositivo (Android Motion Sensors, 2014)

coordenadas são representados em relação a um dispositivo móvel. Para este experimento, o cálculo de velocidade será efetuado através da força aplicada no eixo x do dispositivo.

Com os dados discutidos neste presente tópico, é possível entender como os componentes Bluetooth e de aceleração funcionam em uma aplicação Android. No tópico a seguir será discutido como a aplicação Android e a aplicação Java funcionará no sistema.

2.3. MÉTODOS

Smart grid consiste em uma rede de energia dotada de inteligência. A rede de energia interage com uma infraestrutura de informação, tornando possível o gerenciamento da demanda de energia através de uma rede sistemas de comunicação. Nos trabalhos (Conti et al, 2011) e (Menezes, 2013) é apresentado o problema relacionado a distribuição uniforme de energia das concessionárias no abastecimento de veículos elétricos em ambientes Smart grid. Para a solução deste problema foi proposto sistema de monitoramento de veículos para o controle de velocidade do mesmo. Neste tópico é apresentado um experimento prático agregando todo conhecimento explorado neste artigo para simular e exemplificar um sistema de monitoramento usando uma aplicação Java e um aplicativo Android que se comunicam via protocolo Bluetooth. Como pode ser visto na figura 5, o sistema de monitoramento Bluetooth consiste em um *master*, representado pelo ponto de monitoramento, e pelo *slave*, representado pelo veículo.

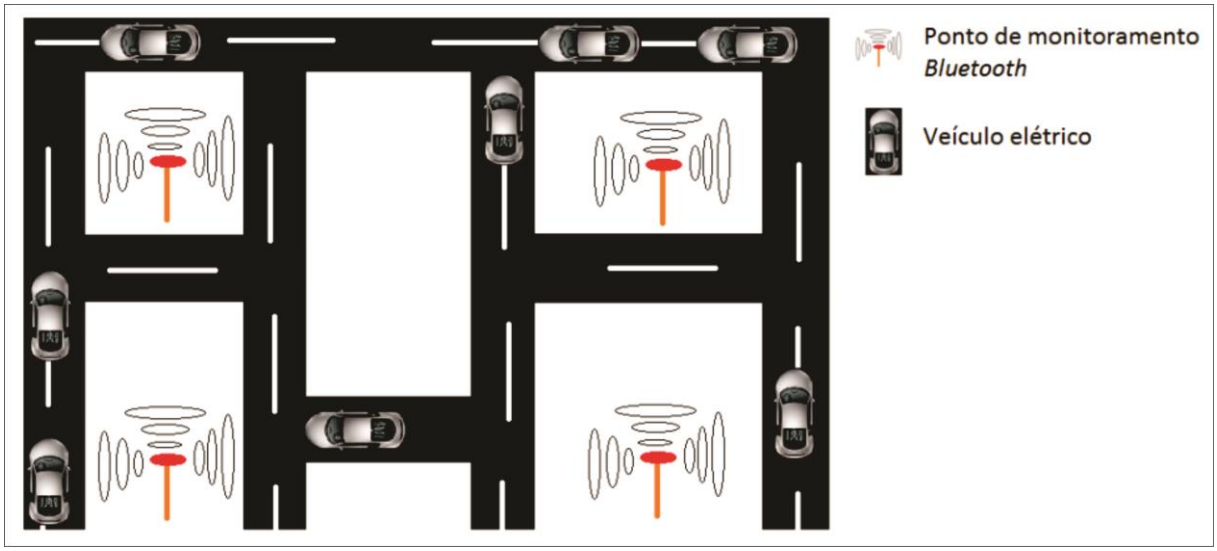

Figura 5: Sistema de monitoramento de veículos.

A aplicação Java tem como objetivo representar o ponto de monitoramento Bluetooth ilustrado na figura 5, enquanto a aplicação Android executada em um *smartphone* representa os veículos. Em linhas gerais, a aplicação *master* estará em um estado de solicitação aguardando que um dispositivo *slave* (dipositivo Android) entre em se raio de alcance. Quando esta aproximação ocorre entre os dispositivos, o dispositivo Android conecta-se automaticamente na aplicação *master*, tornando possível que o *master* possa monitorar o tráfego representado pelo *slave*. Logo após inicializada a conexão entre ambos, em um determinado ponto do tráfego, a aplicação *slave* deverá enviar para a aplicação *master* um arquivo de extensão txt contendo sua velocidade atual. Esta velocidade é calculada usando a aceleração do dispositivo móvel e em um determinado ponto esta velocidade é capturada para um arquivo txt e enviada para a aplicação máster, como já foi dito anteriormente.

2.3.1. Estabelecendo conexão com o master (Aplicação Java)

- Passo 1: Para inicializar um sistema de comunicação *piconet* entre os dispositivos citados neste artigo, a aplicação *master* começa abrindo uma conexão. Para que a inicialização da conexão Bluetooth seja possível, são utilizadas as classes da biblioteca *Bluecove*: UUID, *Connector* e *StreamConnectionNotifier*. Na figura 6 a classe *StreamConnectionNotifier* é responsável consistir em uma abertura de conexão. Ela recebe uma classe *Connection* abre a conexão usando seu método open que recebe como argumento o um endereço de servidor junto do o código de criptografia UUID.

```
StreamConnectionNotifier streamConnNotifier = (StreamConnectionNotifier) Connector
        .open(connectionString);
```

Figura 6: Inicialização da conexão Bluetooth.

-Passo 2: Com a conexão aberta, a classe *StreamConnection* aguardar a solicitação de um *slave* para início da comunicação. Na figura 7, um objeto *StreamConnection* recebe um objeto *StreamConnectionNotifier* que chamará seu respectivo método *acceptAndOpen*, sendo este método responsável por receber um *slave* e começar a comunicação entre os dispositivos.

```
StreamConnection connection = streamConnNotifier.acceptAndOpen();
```

Figura 7: Começando uma conexão com um dispositivo slave.

-Passo 3: Tendo os dispositivos conectados, A classes *RemoteDevice* consistirá no dispositivo *slave* recebendo um objeto *StreamConnection*, como ilustra na figura 8. Para o tratamento do fluxo de dados, o objeto *StreamConnection* será usado para a configuração dos fluxos de entrada de dados, sendo responsável por receber os arquivos txt da aplicação cliente.

```
RemoteDevice dispositivo = RemoteDevice.getRemoteDevice(conexao);
```

Figura 8: Objeto do dispositivo slave.

2.3.2. Estabelecendo conexão com o slave (Aplicação Android)

-Passo 1: O princípio do funcionamento Bluetooth em um dispositivo Android está presente na classe *BluetoothAdaptor*. Esta classe representa o rádio Bluetooth e será o primeiro objeto executado no aplicativo *slave*. Como ilustrado na figura 11.

```
BluetoothAdapter adaptador = BluetoothAdapter.getDefaultAdapter();
```

Figura 9: Configuração do adaptador Bluetooth no Android.

-Passo 2: Logo após, o adaptador Bluetooth estar configurado na aplicação, a mesma vai começar uma sessão de pesquisa por outros dispositivos móveis com o objetivo claro de encontrar a aplicação *master* para começar a comunicação. A classe responsável por fazer a procura por outros dispositivos é o *BluetoothAdaptor* e será o seu método *startDicovery* responsável por encontrar a aplicação *master*.

-Passo 3: Com a aplicação *master* descoberta, o *slave* utilizará do endereço MAC do *master* para solicitar a abertura de conexão. A conexão será feita utilizando a classe BluetoothDevice e *BluetoothSocket*. A classe *BluetoothDevice* consistirá na aplicação *master* e será instanciada usando o seu respectivo endereço MAC. E a classe *BluetoothSocket* é responsável pela conexão com o *master*, como ilustrado na figura 10. Um objeto *BluetoothSocket* utiliza a classe *BluetoothDevice* e uma chave criptográfica UUID para poder ser instanciado.

```
BluetoothSocket socket = device.createRfcommSocketToServiceRecord(UUID
        .fromString("00001101-0000-1000-8000-00805F9B34FB"));
```

Figura 10: Configuração do BluetoothSocket.

Tendo o *BluetoothSocket*, seu método *conect* é usado para começar a comunicação com a aplicação Java *master*.

-Passo 4: Com os passos anteriores finalizados, é possível a troca de dados entre ambas aplicações. A classe *BluetoothSocket* é responsável pela configuração do fluxo de dados que será enviado para o outro dispositivo da comunicação, sendo estes dados um arquivo txt com o valor da velocidade do smartphone Android.

3. RESULTADO DA APLICAÇÃO

O resultado geral da aplicação pode ser visto na figura 13, onde todo o ecossistema da aplicação *master* esta exemplificado em detalhes.

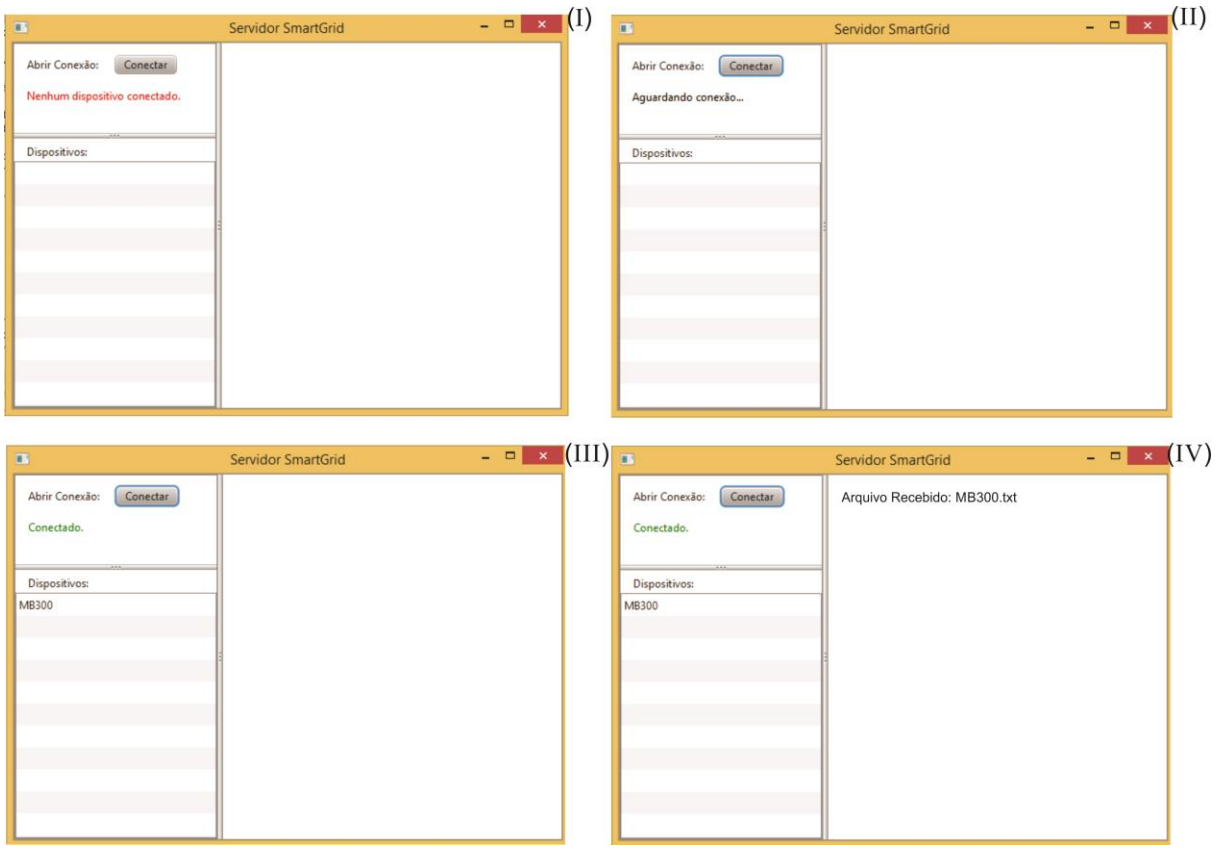

Figura 11: Telas da aplicação master.

A interface da aplicação Java consiste em um botão com o no evento de abrir uma conexão, em uma lista responsável por exibir os dispositivos *slaves* conectados e um bloco em branco onde serão exibidos os arquivos recebidos na tela. O funcionamento da aplicação é dividido em quatro etapas, sendo que cada uma delas está exemplificada na figura 11.

Na primeira etapa de execução, a interface ilustrada na figura 11 (I) é apresentada ao usuário, aguardando apenas que o botão conectar seja clicado. Quando o evento do botão Conectar é ativado, a aplicação cria uma conexão *piconet* aguardando uma solicitação de um *slave*, como pode ser vista na figura 11 (II). No momento que um dispositivo *slave* se conecta a aplicação, a interface mostra seu respectivo nome na lista, podendo ser visto na figura 11 (III). E quando o dispositivo Android passa pelo ponto de monitoramento, o mesmo captura sua velocidade atual baseado na aceleração disponibilizada pelo sensor acelerômetro, cria um arquivo txt com a velocidade e envia o arquivo para a aplicação *master*, como é exemplificada na figura 11 (IV). Por fim, todas quatros etapas abordadas anteriormente podem ser repetidas no momento que outro aplicativo *slave* se conecta ao *master*.

4. CONSIDERAÇÕES FINAIS

Com o resultado obtido neste trabalho é possível simular como os pontos de monitoramento deverão estar posicionados em relação a velocidade média que os veículos deverão estar para que possa ser feita uma comunicação mais eficiente e confiável. A plataforma Android mostrou ser uma alternativa ideal pelo fato de ter os recursos necessários para o controle de tráfego e também por já estar sendo implementadas em veículos no mundo real como é apresentado na reportagem (Wall Street Journal, 2014).

Decorrente as tecnologias utilizadas neste trabalho, a proposta aqui apresentada pode ser implementada em ambientes acadêmicos por possuir ferramentas gratuitas, de fácil implementação e por fim, ser amplamente utilizada na academia por possuir caráter educacional por tratar de ferramentas fundamentais no ensino de engenharia.

5. REFERÊNCIAS / CITAÇÕES

Android Bluetooth. **Guia do Desenvolvedor Adroid**. Disponível em: <http://elinux.org/Android_Architecture/> Acesso 10 de Abril de 2014.

Android Motion Sensors. **Guia do Desenvolvedor Android**. Disponível em: <http://developer.android.com/guide/topics/sensors/sensors_motion.html> Acesso 11 de Abril de 2014.

Bluecove. **Documentação Bluecove**. Disponível em: <http://bluecove.org/> Acesso 20 de Abril de 2014.

CONTI, M.; FEDELI, Dario.; VIRGULTI, M. B4V2G: Bluetooth for electric vehicle to smart grid connection. Dipartimento di Ingegneria Biomedica, Elettronica e Telecomunicazioni Università Politecnica delle Marche, Ancona, Italy, 2011.

LOPES, Y. et al. Minicurso para o SBrT'2012: Smart Grid e IEC 61850: Novos Desafios em Redes e Telecomunicações para o Sistema Elétrico. Departamento de Engenharia de Telecomunicações -– Universidade Federal Fluminense (UFF).Niterói, RJ - Brasil, 2012. Núcleo de Pesquisa em Redes e Computação -– Universidade Salvador (UNIFACS). Salvador, BA - Brasil, 2012.

Menezes, A. L.; Estudo da Viabilidade de Aplicação da Tecnologia Bluetooth na Criação de Ambientes Computacionais Ubíquos. Cascavel, Paraná: Universidade Estadual do Oeste do Paraná, Dezembro, 2008. Monografia.

Oracle Javafx. **Documentação Oracle**. Disponível em: <http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm> Acesso 25 de Abril de 2014.

Wall Street Journal. Disponível em: <http://online.wsj.com/news/article_email/SB10001424052702304591604579288670734733740-lMyQjAxMTAzMDIwOTEyNDkyWj> Acesso 15 de Abril de 2014.

DEVELOPMENT OF A JAVA AND ANDROID APPLICATION FOR SIMULATING TRAFFIC MONITORING SYSTEMS IN SMART GRID

Abstract: *This paper shows the development of a Java Desktop application that can communicate with an Android smartphone using Bluetooth technology as the communication protocol. The present application discloses a system piconet communication, with the Java application as master and as slave Android application. On issues related to a Smart Grid ecosystem, this paper presents a solution to improve the monitoring of electric utilities in energy supply vehicles, enabling greater control of the peak load on the electrical network. In general, the Java application will act as a point of monitoring and Bluetooth Android application as an electric vehicle. When the Android device pass through a given point tracking it sends a file of type txt to master the application containing your current speed. With the data obtained in this work can join their use in classrooms in the school due to the ecosystem of Android platform coming to work with concepts used in computer engineering, and physics studies, networks and programming.*

Key-words: *Android, Bluetooth, Java, Smart Grid*