

**RONALDO DOS SANTOS PEREIRA**

**CONTROLE E AQUISIÇÃO DE DADOS EXPERIMENTAIS COM  
TECNOLOGIA *BLUETOOTH* EM DISPOSITIVOS MÓVEIS**

**RONALDO DOS SANTOS PEREIRA**

**CONTROLE E AQUISIÇÃO DE DADOS EXPERIMENTAIS COM  
TECNOLOGIA *BLUETOOTH* EM DISPOSITIVOS MÓVEIS**

Dissertação apresentada à Faculdade de Engenharia do Câmpus de Ilha Solteira - UNESP, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.

Especialidade: Automação.

Prof. Dr. Luis Carlos Origa de Oliveira

**Orientador**

Ilha Solteira

2016

FICHA CATALOGRÁFICA

Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

Pereira, Ronaldo dos Santos.

P436c      Controle e aquisição de dados experimentais com tecnologia bluetooth em dispositivos móveis / Ronaldo dos Santos Pereira. -- Ilha Solteira: [s.n.],2016  
113 f. : il.

Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2016

Orientador: Luis Carlos Origa de Oliveira  
Inclui bibliografia

1. Android. 2. Google. 3. Tecnologia móvel. 4. Bluetoothg. 5. Smartphone. 6. Appaquisbluet.

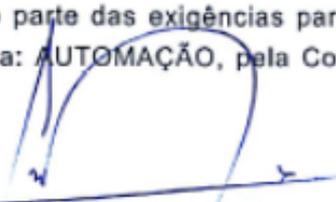
**CERTIFICADO DE APROVAÇÃO**

**TÍTULO DA DISSERTAÇÃO:** Controle e Aquisição de Dados Experimentais com Tecnologia Bluetooth em Dispositivos Móveis

**AUTOR:** RONALDO DOS SANTOS PEREIRA

**ORIENTADOR:** LUIS CARLOS ORIGA DE OLIVEIRA

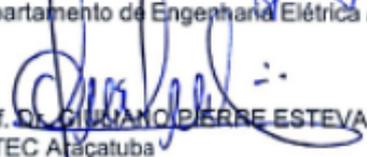
Aprovado como parte das exigências para obtenção do Título de Mestre em ENGENHARIA ELÉTRICA, área: AUTOMAÇÃO, pela Comissão Examinadora:



Prof. Dr. LUIS CARLOS ORIGA DE OLIVEIRA  
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira



Prof. Dr. SERGIO AZEVEDO DE OLIVEIRA  
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira



Prof. Dr. CLEVIANO PIERRE ESTEVAM  
FATEC Aracatuba

Ilha Solteira, 05 de abril de 2016

## DEDICATÓRIA

*À minha querida esposa, Edileusa Gimenes Moralis, por toda compreensão e carinho.*

*À minha filha, Yasmim de Paula Pereira, minha paixão.*

*E aos doutores da minha vida, meus pais:*

*Heminio Lopes Pereira e*

*Aparecida dos Santos Pereira.*

## **AGRADECIMENTOS**

Primeiramente agradeço a Deus por dar-me forças em todos os momentos.

À minha família que sempre me apoiou em toda minha formação, pensando unicamente no meu bem.

Agradeço ao meu orientador, Prof. Dr. Luis Carlos Origa de Oliveira, pela confiança depositada em minha pessoa.

À CAPES/DS (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior/ Programa de Demanda Social - DS), pelo apoio financeiro.

Agradeço ao meu padrinho, Toni Amorim, Rodrigo A. N. de Oliveira e ao Jean Vitor por toda ajuda concedida nos momentos de desesperos.

Agradeço aos meus amigos de república, pelo companheirismo em especial a Fabrício Marques (Doug).

*“Aquele que pensa que tudo sabe  
nada aprende”.*

**Sócrates**

## RESUMO

Os dispositivos móveis portáteis vêm ganhando mercado e se consolidando como uma das ferramentas mais utilizadas no cotidiano, corporativo ou convencional, dados os investimentos das grandes empresas de telefonia móvel, rede de comunicação, empresas de software, *hardware*, entre outras. Diante desse quadro, o presente trabalho abordou a tecnologia móvel em sua evolução, observando como os celulares se transformaram em *smartphones* capazes de gerenciar quase tudo da palma da mão, em razão de sistemas operacionais próprios, sofisticados e com plataformas específicas para desenvolvimento de aplicativos. Tal constatação direcionou esse estudo para o desenvolvimento do aplicativo AppAquisBluet, o qual utilizou a plataforma de desenvolvimento Android, que detêm 75% do mercado móvel, liderada pela Google e seus aliados. Além disso, trata-se de uma plataforma moderna, flexível e *open source*, possuidora de ótima relação custo/benefício, quando comparada a outras plataformas, ou seja, ideal para a implementação do aplicativo AppAquisBluet. Nesse sentido, foram utilizados os métodos e técnicas destinados à plataforma Android, desde a SDK (*Software Development Kit*), linguagem de programação Java, API *bluetooth*, bibliotecas de terceiros como a *GraphView*, utilizada nas implementações gráficas do projeto, sem contar com o eficiente banco de dados *SQLite* destinado a aplicações móveis. Como produto resultante, apresentou-se o aplicativo AppAquisBluet e suas funcionalidades básicas, com a função de realizar aquisições de dados em tempo real, haja vista que esses dados podem ser apresentados em arquivo de texto ou graficamente e armazenados em banco de dados para futuras análises. Para tal aquisição, utilizou-se o hardware condicionador de sinal via *bluetooth* desenvolvido para converter o sinal analógico, captado pelo circuito da ponte de Wheatstone, para digital e transmitido ao aplicativo AppAquisBluet, usando a comunicação *bluetooth*, plotando os dados no visor do aplicativo, além de exportar para a memória de armazenamento do *smartphone* como arquivo de texto para serem tratados, utilizando o software Excel.

**Palavras-chave:** Android. Google. Tecnologia Móvel. *Bluetooth*. *Smartphone*. AppAquisBluet.

## ABSTRACT

The handheld mobile devices are gaining market and consolidating its position as one of the tools most commonly used in daily life of corporations or common users. It is due to investments of the mobile companies, networking companies, software and hardware companies and so on. This work addressed the evolution of mobile technology. We observed how mobile phones have turned into smartphones able to manage almost everything by our hands. Of course, because of sophisticated operational systems and specific platforms for development of applications. The aim of this study was directed to the development of AppAquisBluet application. It used the Android development platform that holds 75% of the mobile market by Google and its allies. Further, it is a modern platform, flexible, open source, an excellent cost and benefit compared to other platforms. That is why it is perfect for the implementation of the AppAquisBluet application. Accordingly, the methods and techniques used in its construction were the same of the Android platform, including the SDK (Software Development Kit), the Java programming language, the Bluetooth API, third-party libraries like GraphView, used in the graphic implementation of the project, and the efficient SQLite database designed for mobile applications. The resulting product was the AppAquisBluet application and its basic features, including the function of performing data acquisition in real time. The data can be coded in a text file or graphically and stored in a database for future analysis. For this acquisition, we used the signal conditioner hardware via Bluetooth. It was developed to convert to digital signal the analog signal received by Weatstone Bridge and then the digital signal could be transmitted to the AppAquisBluet application, using Bluetooth communication by plotting the data in the application display, and exporting to the smartphone storage memory as a text file that can be treated using the Excel software.

**Keywords:** Android. Google. Mobile technology. Bluetooth. Smartphone. AppAquisBluet.

## LISTA DE FIGURAS

Figura 1	A quarta e terceira geração das redes sem fio: benefícios e evolução.....	26
Figura 2	Instrumento de informação conectados à internet.....	27
Figura 3	Classificação das redes sem fio .....	29
Figura 4	Infraestrutura de uma rede <i>ad-hoc</i> .....	31
Figura 5	Os sistemas operacionais mais utilizados na atualidade .....	35
Figura 6	<i>Download</i> do Android SDK .....	43
Figura 7	Emulador do Android executando o AppAquisBluet .....	44
Figura 8	Primeira tela do Eclipse .....	45
Figura 9	Arquitetura geral da plataforma Android .....	47
Figura 10	Ciclo de vida da <i>Activity</i> do Android .....	51
Figura 11	Estrutura do projeto na IDE .....	55
Figura 12	<i>Activity</i> e <i>Intent-filter</i> implementado no <i>AndroidManifest.xml</i> do projeto.....	57
Figura 13	Ilustração de <i>piconet</i> e <i>scatternet</i> .....	59
Figura 14	Permissão <i>Bluetooth</i> no <i>AndroidManifest</i> .....	61
Figura 15	Código para verificar se o <i>Bluetooth</i> está ativo .....	61
Figura 16	Código de solicitação da ativação do <i>Bluetooth</i> .....	62
Figura 17	Diagrama Caso de Uso .....	68
Figura 18	Diagrama de Classe do pacote <b>com.aquisblues</b> .....	69
Figura 19	Diagrama de Classe do pacote <b>com.aquisblues.grafico</b> .....	70
Figura 20	Diagrama de Classe do pacote <b>com.aquisblues.banco</b> .....	71
Figura 21	Diagrama Fluxo de Telas .....	72
Figura 22	Estrutura do Banco de dados .....	73
Figura 23	Telas do aplicativo AppAquisBluet .....	74

Figura 24	Tela de aquisição de dados .....	76
Figura 25	Tela de pesquisa no banco .....	77
Figura 26	Aquisição de dados gráficos .....	79
Figura 27	Busca no banco de dados .....	79
Figura 28	Resultado da nova pesquisa em banco .....	80
Figura 29	Interface do programa de terminal OC-Console .....	81
Figura 30	Configuração do terminal e transferência de pasta .....	82
Figura 31	Comunicação <i>Bluetooth</i> utilizando parâmetros da classe 2 .....	82
Figura 32	Módulo <i>Bluetooth</i> HC-06 utilizado no projeto .....	83
Figura 33	Ponte de <i>Wheatstone</i> .....	85
Figura 34	Dois tipos básicos de extensômetros .....	87
Figura 35	Circuito elétrico do extensômetro com a ponte de <i>Wheatstone</i> .....	88
Figura 36	Diagrama de blocos do receptor e condicionador de sinal .....	89
Figura 37	Condicionador de sinal via <i>Bluetooth</i> .....	90
Figura 38	Fator de escala .....	93
Figura 39	Componentes do condicionador de sinal .....	94
Figura 40	Aplicação recebendo dados .....	95
Figura 41	Ensaio com pesos aleatórios .....	96
Figura 42	Ensaio com pesos aleatórios tratado no software Excel .....	97
Figura 43	Ensaio com pesos aleatórios apresentando dados do tipo <b>texto</b> .....	98
Figura 44	Resultado da <b>Arq Dados</b> representado no software Excel .....	99
Figura 45	Resultado gráfico do <b>Peso TG</b> .....	100
Figura 46	Resultado em carga e descarga dos valores apresentado na tabela 7.....	101
Figura 47	Resultado gráfico dos valores de carga e descarga.....	102
Figura 48	<b>Arquivo de texto</b> salvo na memória de armazenamento do <i>smartphone</i> .....	102
Figura 49	Arquivo do tipo <b>texto</b> apresentado pelo aplicativo AppAquisBluet.....	103

Figura 50	Arquivo do tipo <b>texto</b> relacionado ao dado <b>Peso TD</b> .....	104
Figura 51	Arquivo do tipo <b>texto</b> relacionado ao dado <b>Peso TD</b> de forma contínua.. . . .	105

## LISTA DE TABELAS

Tabela 1	Tecnologia aplicada em celular digital e PCS .....	28
Tabela 2	Sistemas operacionais móveis .....	32
Tabela 3	Sistema operacional móvel mais utilizado na atualidade .....	33
Tabela 4	Ambiente de desenvolvimento .....	39
Tabela 5	Simulação das aplicações .....	40
Tabela 6	Custo para desenvolvimento e distribuição de aplicações .....	41
Tabela 7	Tabela de calibragem do transdutor de pesagem .....	92

## LISTA DE ABREVIATURAS E SIGLAS

AIEE	<i>American Institute of Electrical Engineers</i>
ADT	<i>Android Development Tools</i>
AM	<i>Amplitude Modulation</i>
ANATEL	<i>Agencia Nacional de Telecomunicações</i>
ASCII	<i>American Standard Code for Information Interchange</i>
APIs	<i>Application Programming Interface</i>
AMPS	<i>Advanced Mobile Phone System</i>
ASF	<i>Apache Software Foundation</i>
CDMA	<i>Code Division Multiple Access</i>
DDL	<i>Data Definition Language</i>
DEX	<i>Dalvik Executable</i>
DML	<i>Data Manipulation Language</i>
FHS	<i>Frequency Hopping Synchronization</i>
FM	<i>Frequency-Modulation</i>
GEO	<i>Geosynchronous Earth Orbit</i>
GCC	<i>GNU Compiler Collection</i>
GDB	<i>GNU Debugger</i>
GSM	<i>Global System for Mobile Communications</i>
HP	<i>Hewlett-Packard</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
IDE	<i>Integrated Development Environment</i>
IRE	<i>Institute of Radio Engineers</i>
JDK	<i>Java Development Kit</i>
JRE	<i>Java Runtime Environment</i>
JVM	<i>Java Virtual Machine</i>
LEO	<i>Low Earth Orbit</i>
LTE	<i>Long Term Evolution</i>
LLVM	<i>Low-Level Virtual Machine</i>
MEO	<i>Medium Earth Orbit</i>
NTT	<i>Nippon Telegraph e Telephone</i>
OHA	<i>Open Handset Alliance</i>
PDA	<i>Personal Digital Assistant</i>
PCD	<i>Personal Digital Communications</i>
PCS	<i>Personal Communications Services</i>
RDX	<i>Recepção de Dados</i>
RIM	<i>Research In Motion</i>

RFCOMM	<i>Radio frequency communications</i>
SDK	<i>Software Development Kit</i>
SDP	<i>Service Discovery Protocol</i>
SMS	<i>Short Message Service</i>
SQL	<i>Structured Query Language</i>
TDMA	<i>Time Division Multiple Access</i>
TXD	<i>Transmissão de Dados</i>
UMTS	<i>Universal Mobile Telecommunications</i>
UUID	<i>Universally Unique Identifier</i>
WAP	<i>Wireless Application Protocol</i>
WIMAX	<i>Worldwide Interoperability for Microwave Access</i>
WLAN	<i>Wireless Local Area Network</i>
WMAN	<i>Wireless Metropolitan Area Network</i>
WPAN	<i>Wireless Personal Area Network</i>
WWAN	<i>Wireless Wide Area Network</i>
WWDC	<i>Apple Worldwide Developers Conference</i>

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO</b>	<b>20</b>
1.1	ESTRUTURA DO TRABALHO	21
1.2	OBJETIVO GERAL	22
1.3	OBJETIVOS ESPECÍFICOS	22
1.4	METODOLOGIA E PROPOSTA DO TRABALHO	22
<b>2</b>	<b>TECNOLOGIA DA COMPUTAÇÃO MÓVEL COM ÊNFASE EM DISPOSITIVOS MÓVEIS PORTÁTEIS</b>	<b>24</b>
2.1	HISTÓRICO DA EVOLUÇÃO DA COMPUTAÇÃO MÓVEL	24
2.2	CARACTERÍSTICAS DA COMPUTAÇÃO MÓVEL	24
2.3	COMUNICAÇÃO MÓVEL	26
2.4	SISTEMA DE COMUNICAÇÃO PESSOAL	27
2.5	COMUNICAÇÃO VIA SATÉLITE	28
2.6	REDES LOCAIS SEM FIOS	29
2.7	SISTEMAS OPERACIONAIS PARA DISPOSITIVOS MÓVEIS	31
2.8	CONSIDERAÇÕES FINAIS DO CAPÍTULO	35
<b>3</b>	<b>CENÁRIO DA TECNOLOGIA MÓVEL ABORDADA NO DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS</b>	<b>37</b>
3.1	DISTINÇÃO ENTRE AS PLATAFORMAS iOS E ANDROID	38
3.2	AMBIENTE DE DESENVOLVIMENTO	39
3.3	COMUNIDADE DE DESENVOLVIMENTO	39
3.4	SIMULAÇÃO DAS APLICAÇÕES	39
3.5	CUSTO DE DESENVOLVIMENTO E DISTRIBUIÇÃO DE APLICAÇÕES ANDROID E iOS	40
3.6	CONSIDERAÇÕES FINAIS DO CAPÍTULO	41

<b>4</b>	<b>PLATAFORMA DE DESENVOLVIMENTO DE APLICAÇÕES PARA DÍSPPOSITIVOS MÓVEIS: GOOGLE ANDROID</b>	<b>42</b>
4.1	FERRAMENTAS QUE CONSTITUEM A PLATAFORMA GOOGLE ANDROID	42
4.1.1	<b>Android SDK</b>	<b>42</b>
4.1.1.1	<i>Emulador</i>	43
4.1.1.2	<i>Principais APIs</i>	44
4.1.2	<b>JDK (Java Development Kit)</b>	<b>45</b>
4.1.3	<b>Eclipse</b>	<b>45</b>
4.1.4	<i>Android Developer Tools (ADT)</i>	46
4.2	ARQUITETURA DO ANDROID	46
4.2.1	<b>Aplicações</b>	<b>47</b>
4.2.2	<i>Application Framework</i>	47
4.2.3	<b>Bibliotecas</b>	<b>48</b>
4.2.4	<i>Android Runtime</i>	49
4.2.5	<i>Linux Kernel</i>	50
4.3	CICLO DE VIDA DA <i>ACTIVITY</i> NO ANDROID	50
4.4	CARACTERÍSTICAS PRÓPRIAS DA PLATAFORMA ANDROID	52
4.4.1	<b>Java</b>	<b>53</b>
4.4.2	<b>Google Play</b>	<b>53</b>
4.4.3	<b>Código Aberto e Livre</b>	<b>53</b>
4.5	CONSIDERAÇÕES FINAIS DO CAPÍTULO	53
<b>5</b>	<b>MÉTODOS UTILIZADOS NO DESENVOLVIMENTO DO APPAQUISBLUET</b>	<b>55</b>
5.1	ESTRUTURA DO PROJETO APPAQUISBLUET NA IDE ECLIPSE	55
5.1.1	<i>Activity</i>	56
5.1.2	<i>Intent</i>	57
5.1.3	<i>Broadcast Receiver</i>	57

5.1.4	<i>Service</i>	58
5.2	<i>BLUETOOTH</i>	58
5.3	<i>API BLUETOOTH PARA ANDROID</i>	60
5.3.1	<b>Tornando Visível o Dispositivo Remoto para Liberar a Comunicação com o Adaptador <i>Bluetooth</i></b>	<b>62</b>
5.3.2	<b>Estabelecendo Canais RFCOMM na Comunicação Bluetooth</b>	<b>63</b>
5.4	<i>BIBLIOTECA GRAPHVIEW</i>	64
5.5	<i>BANCO DE DADOS</i>	65
5.5.1	<b>Elementos Básicos de um Banco de Dados</b>	<b>65</b>
5.5.2	<b>Modelo de Dados</b>	<b>66</b>
5.5.3	<b>Linguagem de Banco de Dados</b>	<b>66</b>
5.5.4	<b>Banco de Dados <i>SQLite</i></b>	<b>67</b>
5.6	<i>MODELAGEM DA APLICAÇÃO APPAQUISBLUET</i>	67
5.6.1	<b>Diagrama de Caso de Uso</b>	<b>68</b>
5.6.2	<b>Diagrama de Classes</b>	<b>68</b>
5.6.3	<b>Diagrama de Fluxo das Telas</b>	<b>71</b>
5.6.4	<b>Estrutura do Banco de Dados</b>	<b>72</b>
5.7	<i>CONSIDERAÇÕES FINAIS DO CAPÍTULO</i>	73
6	<b>RESULTADO DOS MÉTODOS UTILIZADOS E AQUISIÇÃO DE DADOS VIA BLUETOOTH</b>	<b>74</b>
6.1	<i>APPAQUISBLUET</i>	74
6.1.1	<b>Resultado do Pareamento Entre o Dispositivo e o Módulo <i>Bluetooth</i></b>	<b>75</b>
6.1.2	<b>Tela de Aquisição de Dados em Tempo Real</b>	<b>75</b>
6.1.3	<b>Pesquisando Dados no Banco</b>	<b>77</b>
6.1.3.1	<i>Pesquisar dados gráficos em banco</i>	78
6.2	<i>TECNOLOGIAS UTILIZADAS NA AQUISIÇÃO DE DADOS</i>	81

<b>6.2.1</b>	<b>Software OC-Console</b>	<b>81</b>
<b>6.2.2</b>	<b>Estabelecendo a Comunicação</b>	<b>82</b>
<b>6.2.3</b>	<b>Módulo <i>Bluetooth</i> HC-05</b>	<b>83</b>
<b>6.3</b>	<b>CONSIDERAÇÕES FINAIS DO CAPÍTULO</b>	<b>83</b>
<b>7</b>	<b>ENSAIOS UTILIZANDO O APLICATIVO APPAQUISBLUET E PONTE DE WHEATSTONE INTERMEDIADA PELA COMUNICAÇÃO <i>BLUETOOTH</i></b>	<b>84</b>
<b>7.1</b>	<b>PONTE DE WHEATSTONE</b>	<b>84</b>
<b>7.1.1</b>	<b>Fazendo-se Uso do Extensômetro</b>	<b>86</b>
<b>7.2</b>	<b>CONDICIONADOR DE SINAL COM TRANSMISSÃO DE DADOS VIA <i>BLUETOOTH</i> COM PONTE DE <i>WHEATSTONE</i></b>	<b>88</b>
<b>7.2.1</b>	<b>Condicionador de Sinal com Extensômetro Fixado na Barra de Pesagem</b>	<b>89</b>
<b>7.2.2</b>	<b>Formato dos Dados do Condicionador de Sinal</b>	<b>91</b>
<b>7.2.3</b>	<b>Calibração do Transdutor de Pesagem</b>	<b>91</b>
<b>7.3</b>	<b>RESULTADOS DA AQUISIÇÃO DE DADOS REALIZADA COM O CONDICIONADOR DE SINAL VIA <i>BLUETOOTH</i> UTILIZANDO A BARRA DE PESAGEM COM PONTE DE <i>WEATSTONE</i></b>	<b>94</b>
<b>7.3.1</b>	<b>Procedimento para Realizar os Ensaios</b>	<b>95</b>
<b>7.3.2</b>	<b>Aquisição de Dados Exibida em Texto</b>	<b>98</b>
<b>7.3.3</b>	<b>Ensaios Utilizando Pesos Relacionados à Tabela 7</b>	<b>99</b>
<b>7.3.4</b>	<b>Ensaios Utilizando Peso Relacionado à Tabela 7 no Formato Texto</b>	<b>103</b>
<b>7.4</b>	<b>CONSIDERAÇÕES FINAIS DO CAPÍTULO</b>	<b>105</b>
<b>8</b>	<b>CONCLUSÕES</b>	<b>106</b>
	<b>REFERÊNCIAS</b>	<b>109</b>

## 1 INTRODUÇÃO

Em razão de avanços e investimentos tecnológicos, a relação entre homem e máquina evoluiu consideravelmente, visto que, hoje, em pleno século XXI, a tecnologia móvel, por exemplo, está presente no dia-a-dia das pessoas, oferecendo-lhes não só o conforto da mobilidade dos aparelhos, mas também o que há de mais moderno em tecnologia digital nos diferentes setores do cotidiano delas.

Diante desse quadro, este trabalho discorreu sobre os primeiros dispositivos móveis desenvolvidos na história, a evolução da comunicação sem fio e sua difusão, a exemplo da criação das subáreas da computação móvel, redes de comunicação e as implementações de sistemas em dispositivos móveis, tendo como objetivo o desenvolvimento de uma aplicação móvel capaz de realizar testes de aquisição de dados intermediada pela comunicação *bluetooth*, direcionada a *smartphones* executados especificamente pelo sistema operacional Android.

De acordo com Lecheta (2013), mais de três bilhões de pessoas possuem um aparelho celular, o que corresponde, aproximadamente, a mais da metade da população mundial. Com o crescimento considerável da utilização de celulares e usuários cada vez mais exigentes, houve a necessidade de agradar os dois segmentos do setor, o corporativo e o convencional, de modo que surgiram, ao longo dos anos, empresas especializadas em desenvolvimento de aplicativos móveis com plataformas específicas e tecnologias próprias.

Entre as várias plataformas de desenvolvimento de aplicativos móveis, duas detêm aproximadamente 91% do mercado, a plataforma iOS e plataforma Android, fato que levou este estudo, considerando a relação custo-benefício, a adotar a plataforma Android para o desenvolvimento do projeto AppAquisBluet. Tal plataforma, considerada moderna, flexível e *open source*, baseada no *kernel* do Linux, se tornou líder no mercado de sistema operacional móvel, com mais de 800 mil aplicativos disponibilizados na loja Google *Play Store*, alcançando a marca de 1 milhão, em 2013, detendo, aproximadamente, 75% do mercado de dispositivos móveis, não se limitando apenas a *smartphones* ou *tablets* (QUERINO, 2013).

Nesse sentido, usufruindo de toda tecnologia e ferramenta disponibilizadas na plataforma Android, a estrutura do projeto foi implementada, utilizando a IDE Eclipse ADT Bundle, constituída de várias pastas, arquivos e bibliotecas, a fim de demonstrar a interface do aplicativo e suas funcionalidades, com a execução de testes de aquisição de dados e apresentação dos resultados em arquivo de texto e graficamente por intermédio da biblioteca *GraphView* e armazenamento de tais resultados no banco *SQLite* destinados para *smartphones*.

Por fim, foram realizados ensaios utilizando a aplicação AppAquisBluet juntamente com o hardware condicionador de sinal via *bluetooth* com ponte de *Wheatstone*.

Contexto em que os pesos (em gramas) de variadas massas são colocados na barra de pesagem instrumentada com uma ponte de *Wheatstone* para serem medidos, digitalizados e, em seguida, transmitidos através do adaptador *bluetooth* para o aplicativo AppAquisBluet, visto que os dados recebidos pelo aplicativo podem ser armazenados em banco e exportados para a memória do *smartphone* para futuras análises.

## 1.1 ESTRUTURA DO TRABALHO

Este trabalho possui oito capítulos e está organizado da seguinte forma: no Capítulo 1 apresenta-se a introdução, os objetivos gerais e específicos, bem como a metodologia utilizada e a estrutura do trabalho.

No Capítulo 2, aborda-se a tecnologia da computação móvel com ênfase em dispositivos móveis portáteis, quanto sua característica e evolução.

No Capítulo 3, contextualiza-se o cenário da tecnologia móvel abordada no desenvolvimento de aplicações móveis, com o intuito de apresentar um panorama quanto ao custo/benefício envolvendo as duas plataformas mais utilizadas na atualidade.

No Capítulo 4, discorre-se sobre a plataforma Google Android, quanto ao seu contexto histórico, características próprias, ferramentas utilizadas em seu núcleo, arquitetura da plataforma e sua SDK, que auxilia o desenvolvedor na criação de aplicações cada dia mais sofisticadas.

No Capítulo 5, expõe-se os métodos e técnicas utilizadas no desenvolvimento do aplicativo AppAquisBluet, para que suas funcionalidades respeitem todos os parâmetros estipulados no código fonte.

No Capítulo 6, exibem-se os resultados dos métodos utilizados no desenvolvimento do aplicativo AppAquisBluet e resultados das aquisições de dados via *bluetooth*.

No Capítulo 7, apresentam-se os ensaios com aplicativo AppAquisBluet e o hardware condicionador de sinal via *bluetooth* com ponte de *Wheatstone* intermediada pela comunicação *bluetooth*.

No Capítulo 8, apresenta-se a conclusão do trabalho proposto, abordando os tópicos relacionados no decorrer do seu desenvolvimento, além de trabalhos futuros relacionados ao teste de aquisição de dados envolvendo a aplicação juntamente com a ponte de *Wheatstone*.

## 1.2 OBJETIVO GERAL

Este trabalho propôs analisar a tecnologia móvel com objetivo de desenvolver uma aplicação móvel para realizar testes de aquisição de dados intermediados pela comunicação *bluetooth*, utilizando o hardware condicionador de sinal via *bluetooth* com ponte de Wheatstone, a fim de proporcionar mais segurança ao usuário nas aquisições de dados envolvendo periculosidades.

## 1.3 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho foram:

- Apresentar a tecnologia móvel e sua contribuição para a mobilidade;
- Demonstrar o cenário envolvendo o desenvolvimento de aplicações móveis, com objetivo de analisar a plataforma mais viável para o desenvolvimento do estudo em tela;
- Abordar a plataforma Google Android no seu contexto histórico, o ambiente de desenvolvimento (SDK), arquitetura, características e ferramentas;
- Apresentar os métodos utilizados no desenvolvimento do aplicativo AppAquisBluet que envolve as tecnologias móveis, para se chegar a um produto final;
- Mostrar o aplicativo AppAquisBluet como resultante dos métodos utilizados no decorrer do desenvolvimento deste trabalho e as funcionalidades básicas do aplicativo;
- Apresentar os resultados de aquisição de dados tanto em arquivo de texto ou graficamente, utilizando-se de estruturas específicas do aplicativo, com o armazenamento de tais resultados em banco de dados, além de exportá-los para a memória de armazenamento do *smartphone* para futuras análises;
- E por fim, apresentar resultados de ensaios que envolve o aplicativo AppAquisBluet com o hardware condicionador de sinal via *Bluetooth* com ponte de *Wheatstone*.

## 1.4 METODOLOGIA E PROPOSTA DO TRABALHO

A revisão bibliográfica foi um dos caminhos escolhidos no desenvolvimento desta pesquisa, em que levantamentos bibliográficos relacionados ao tema deste estudo foram realizados, contextualizando a evolução da tecnologia móvel, com a abordagem da plataforma Android, em suas características, arquitetura e ferramentas disponibilizadas, a fim de corroborar com os instrumentos necessários ao desenvolvimento do aplicativo AppAquisBluet, o qual utiliza métodos e técnicas para aquisição de dados em tempo real, intermediada pela comunicação *bluetooth*.

A proposta deste trabalho limitou-se à tecnologia da computação móvel com ênfase em dispositivos móveis portáteis, cenário que envolve o desenvolvimento de aplicações móveis, plataforma Google Android, métodos utilizados no desenvolvimento do AppAquisBluet, resultado desses métodos e aquisição de dados via *Bluetooth* com hardware condicionador de sinal via *bluetooth* com ponte de *Wheatstone*, para demonstrar a potencialidade da aplicação em destaque quanto da tecnologia móvel. Todos esses processos foram utilizados para se chegar a um produto final, obtendo custo muito baixo por se tratar de uma plataforma de desenvolvimento móvel *open source*.

## 2 TECNOLOGIA DA COMPUTAÇÃO MÓVEL COM ÊNFASE EM DISPOSITIVOS MÓVEIS PORTÁTEIS

Neste capítulo, apresenta-se um levantamento bibliográfico voltado à tecnologia da computação móvel com ênfase em dispositivo móveis portáteis. O interesse é apontar quais foram os primeiros dispositivos móveis desenvolvidos na história, a evolução da comunicação sem fio e seus desdobramentos.

### 2.1 HISTÓRICO DA EVOLUÇÃO DA COMPUTAÇÃO MÓVEL

Entre as influências sofridas pelas novas tecnologias, no decorrer do tempo, no contexto da computação móvel, Mateus e Loureiro (1998) destacam algumas: a descoberta de Hans Christian Oersted, no ano de 1820, em que a corrente elétrica produz um campo magnético; o primeiro sistema de comunicação desenvolvido - o telégrafo - na metade do século XIX; a descoberta da radiotelegrafia, no final do século XIX, por Guglielmo Marconi; a invenção do segundo sistema de comunicação - o telefone - por Alexander Graham Bell, e o computador, a terceira geração do sistema de comunicação.

Os desdobramentos dessas diferentes tecnologias propiciaram o desenvolvimento da comunicação sem fio, a qual forma várias subáreas dentro da tecnologia digital, a exemplo da comunicação de celular: comunicação móvel, serviços de comunicação pessoal, comunicação via satélite e redes locais sem fios (MATEUS; LOUREIRO, 1998).

### 2.2 CARACTERÍSTICAS DA COMPUTAÇÃO MÓVEL

Entre as características da computação móvel, tem-se a comunicação via celular, constituída por alguns pontos marcantes com o passar dos anos. Segundo Pellanda (2005), em 1973, nos EUA, Martin Cooper instalou a primeira estação de sinais *radio-base*, provando que os sinais poderiam cobrir áreas onde as pessoas se deslocavam. Consequentemente, na mesma década, no ano de 1979, foi a vez do Japão lançar o primeiro serviço de telefonia celular pela empresa NTT (*Nippon Telegraph e Telephone*). Com a comunicação via celular se tornando um dos maiores meios de comunicação no mundo, foi inevitável a evolução de sua geração, marcada por inovações e sofisticação, como demonstrado por Pellanda (2005):

- A 1ª geração de celulares foi marcada por ser constituída de sistema analógico conhecido como AMPS (*Advanced Mobile Phone System*) estabelecida na Europa e adotada por toda tecnologia de celular no mundo;

- A 2<sup>a</sup> geração de celulares foi implantada em 1990, tendo como principal característica ser totalmente digital, proporcionando a transmissão de dados para navegação na internet em baixa velocidade, por volta de 9 kbps e com alto custo. Outro fator marcante neste mesmo ano foi a divisão de diferentes padrões, como: GSM (*Global System for Mobile Communications*), desenvolvida na Europa; CDMA (*Code Division Multiple Access*) desenvolvida nos Estados Unidos, e PCD (*Personal Digital Communications*) no Japão. Contudo, a sua principal implementação foi o protocolo de SMS (*Short Message Service*) batizado por algumas operadoras brasileiras como “torpedo”, primeiro serviço de dados que se tornou uma importante ferramenta de comunicação. Outro ponto marcante nesta geração foi o surgimento do protocolo WAP (*Wireless Application Protocol*), primeira forma de navegação em páginas na internet intermediada por um celular;
- A 3<sup>a</sup> geração é marcada pela velocidade de transmissão que trabalhava entre 384 kbps e 2 Mbps, podendo oferecer serviços como vídeo conferência, transmissão de áudio e vídeo com qualidade equiparada a DVD. A empresa japonesa DoCoMo foi a primeira no mundo a oferecer os serviços da 3<sup>a</sup> geração, seguida da Inglaterra, implantando novas redes como a UMTS (*Universal Mobile Telecommunications*) e, ao longo dos anos vários, países europeus começaram a operar o sistema. No Brasil, a tecnologia da 3<sup>a</sup> geração só começou a ser operada em algumas cidades no ano de 2005 disponibilizada pela operadora Vivo.
- Nos dias atuais, a 4<sup>a</sup> geração de comunicação vem sendo implantada, contexto em que são oferecidas as mais diferentes funcionalidades em um celular/*smartphone*, ou seja, desde a leitura de um *e-mails*, compras *on-line* a transações bancárias, além de alcançar outras necessidades do usuário, e atingir velocidades de até 50 Mbps.

Segundo relato do diretor do Departamento de Banda Larga do Ministério das Comunicações, Artur Cimbra de Oliveira, coletado pelo Sebrae (2014), durante a última Copa do Mundo FIFA de 2014, “[...] até o início do campeonato, prestadoras de serviços de telecomunicações nas 12 cidades-sede já devem oferecer ao público a nova tecnologia”. Essa ação deixava registrado na história, pela primeira vez, a implantação de uma nova tecnologia de telecomunicação no Brasil ao mesmo tempo em que está era implantada nos países de primeiro mundo. Artur relata que a implantação da telefonia celular 4G no país é um dos legados que a Copa deixará para a sociedade brasileira (SEBRAE, 2014).

## 2.3 COMUNICAÇÃO MÓVEL

Cada vez mais acessível ao usuário final, a comunicação móvel, no Brasil, segundo Pellanda (2009) e dados da ANATEL (*Agencia Nacional de Telecomunicações*), no ano de 2008, registrou mais de 130 milhões de aparelhos usufruindo desse tipo de tecnologia, enquanto, na China, essa marca era de 600 milhões de aparelhos, de acordo com informações do Ministério de Indústria e Tecnologia da Informação da China (PELLANDA, 2009).

Tal crescimento disponibilizou no mercado aparelhos com tecnologia cada vez mais sofisticada, como é o caso dos *smartphones* e computadores pessoais. Paralelamente, têm-se as redes da terceira e quarta geração de celulares que contribuem com esse desenvolvimento, definidas por Gozálvez (2014), como segue:

- WIMAX (*Worldwide Interoperability for Microwave Access*), semelhante ao funcionamento da *Wifi*, porém com abrangência metropolitana;
- LTE (*Long Term Evolution*), tecnologia de telefonia móvel conhecida como 4G.

Na Figura 1, têm-se dois tipos de redes da 4ª geração utilizadas no mundo: as redes LTE (*Long Term Evolution*) e as redes WIMAX (*Worldwide Interoperability for Microwave Access*).

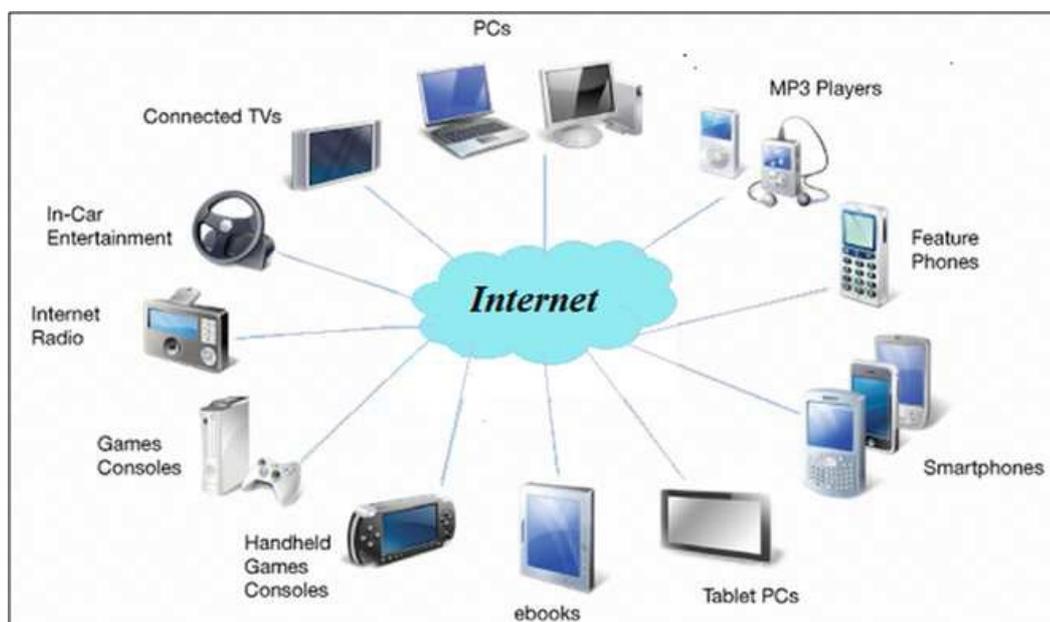
Figura 1 - A quarta e terceira geração das redes sem fio: benefícios e evolução.

Tecnologia Móvel	Velocidade de Banda Larga Móvel
LTE	120 Mbps
Wimax	72 Mbps
3G (HSPA)	14 Mbps

Fonte: (NIKOLOFSKI, 2013).

A evolução dos aparelhos celulares em seus diferentes aspectos, juntamente com a expansão das redes sem fio em vários formatos e abrangências, tornou a comunicação móvel possível, através da conexão dos aparelhos portáteis em rede, permitindo o deslocamento com acesso à internet para qualquer parte do globo terrestre, propiciando ao usuário a potencialidade da comunicação móvel, disponibilizada pela tecnologia (GOZÁLVEZ, 2014). Na Figura 2 apresenta-se alguns instrumentos de informação, utilizando a comunicação móvel conectado à internet.

Figura 2 - Instrumentos de informação conectados à internet.



Fonte: Elaboração do autor.

## 2.4 SISTEMA DE COMUNICAÇÃO PESSOAL

O Sistema de Comunicação Pessoal (Personal Communication System - PCS) foi constituído por três grupos de tecnologias de telefone celular digital: GSM-1900, CDMA IS-95 e TDMA IS-136. Assim, conectando o telefone celular ao PC, o usuário pode interagir com alguns aplicativos como: serviço de e-mail, fax, navegador da internet, acesso a rede corporativa de dados, entre outras funções disponibilizadas pela tecnologia em questão (RATTMANN; RATTMANN, 2014).

Com o avanço da tecnologia, observa-se que os sinais digitais podem ser processados em contraste com o sinal analógico e representado em estado binário de 1 ou 0, *on* ou *off*, *true* ou *false*. Desse modo, como os computadores armazenam dados em estado binário, isso torna os sinais digitais ideais para transmissão e armazenamento de dados (RATTMANN; RATTMANN, 2014).

Segundo Rattmann e Rattmann (2014), "[...] a tecnologia digital é mais espectral que a analógica", contexto em que vários usuários compartilham a mesma frequência ou rádio-canal. Nesse sentido, as redes com alta capacidade de tráfego ficam sobrecarregadas. Entretanto, o sistema dos celulares é baseado no reuso de frequência. Assim, quando uma frequência está ocupada, automaticamente é redirecionado para outro canal que esteja disponível.

Na Tabela 1, têm-se algumas tecnologias aplicadas em celular digital de 800 MHz e PCS 1.900 MHz.

Tabela 1 - Tecnologia aplicada em celular digital e PCS.

<b>Tecnologia</b>	<b>Classificação</b>	<b>Banda de frequência (MHz)</b>
AMPS	Celular Analógico	800
CDMA	Celular Digital e PCS	800 ou 1900
TDMA	Celular Digital e PCS	800 ou 1900
GSM	PCS	1900
GSM	PCS (Brasil e Europa)	1800

Fonte: (RATTMANN; RATTMANN, 2014).

## 2.5 COMUNICAÇÃO VIA SATÉLITE

A comunicação via satélite é uma das tecnologias que compõem a comunicação móvel com características bem peculiar. A principal delas é a alta capacidade e possibilidade de atender uma demanda expressiva de usuários a baixo custo. Além disso, esta tecnologia foi criada para complementar as já existentes no mercado, porém cobrindo regiões remotas não atendidas pelas demais tecnologias. Contudo, o quesito segurança era frágil, já que qualquer unidade receptora poderia captar o sinal enviado pelo satélite, o que levou à utilização da criptografia nesse tipo de comunicação, a fim de garantir maior segurança aos dados (MATEUS; LOUREIRO, 1998).

Mateus e Loureiro (1998) relatam que na tecnologia de comunicação via satélite, têm-se três níveis estabelecidos para a comunicação:

- LEO (*Low Earth Orbit*) são satélites de baixa órbita que estão posicionados em torno de 1.000 km de altitude, em diferentes posições com relação a terra;
- MEO (*Medium Earth Orbit*) são satélites de órbitas médias, instalados aproximadamente a 10.000 km de altitude;
- GEO (*Geosynchronous Earth Orbit*) são satélites de órbitas elevadas ou geostacionárias, e estão situados aproximadamente a 36.000 km de altitude nas regiões próximas a linha do equador.

Na atualidade, além dos satélites do Sistema de Posicionamento Global em órbita, existem os satélites científicos, satélites militares, satélites de comunicação cuja função específica é retransmitir sinais entre pontos distantes da Terra. Gomes (2015), define a função desses satélites como: “[...] servem para retransmitir dados, sinais de televisão, rádio ou mesmo telefone.

Os chamados (telefones por satélites) baseiam-se em uma rede Iridium, rede de satélite de baixa altitude” (GOMES, 2015).

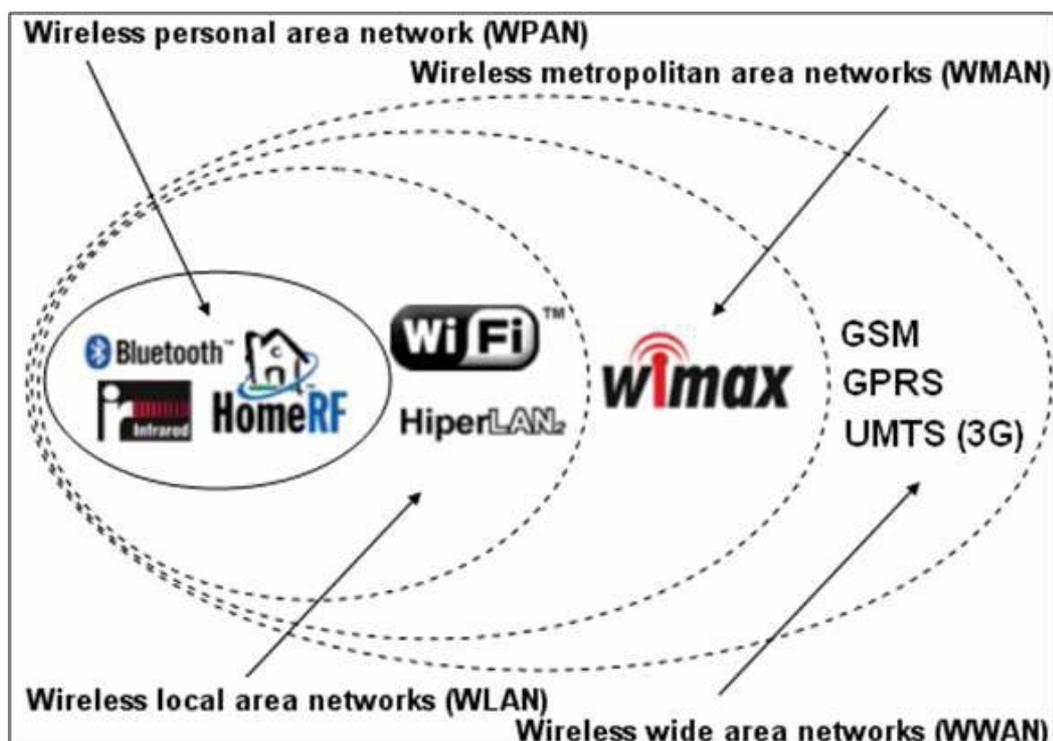
## 2.6 REDES LOCAIS SEM FIOS

Segundo Mateus e Loureiro (1998), "[...] a comunicação sem fio tem sido usada muito antes das redes de celulares, com as emissões via rádio AM e FM, as comunicações navais, e a própria televisão", o que leva a concluir que a grande maioria dos sistemas de comunicação sem fio se baseia na comunicação via rádio e na alocação de frequência, visto que a mesma necessita de alguns fatores que contribuam para sua eficiência, como as antenas, potência de transmissão e relevo ou meios interferentes, garantindo, dessa forma, que o sinal seja transmitido com boa qualidade para o usuário final, ou seja sem interferência de ruídos.

As redes sem fio têm apresentando significativos benefícios para o usuário final, visto que atuam em ambientes domésticos, chamado de WPAN (*Wireless Personal Area Network*), fazendo a comunicação entre os dispositivos dentro de uma residência ou escritório, até as redes de sistemas celulares, conhecida como WWAN (*Wireless Wide Area Network*), passando pelas WLAN (*Wireless Local Area Network*) e as WMAN (*Wireless Metropolitan Area Network*).

Na Figura 3, apresenta-se a classificação das redes sem fio e as tecnologias utilizadas em sua comunicação na transmissão de dados.

Figura 3 - Classificação das redes sem fio.



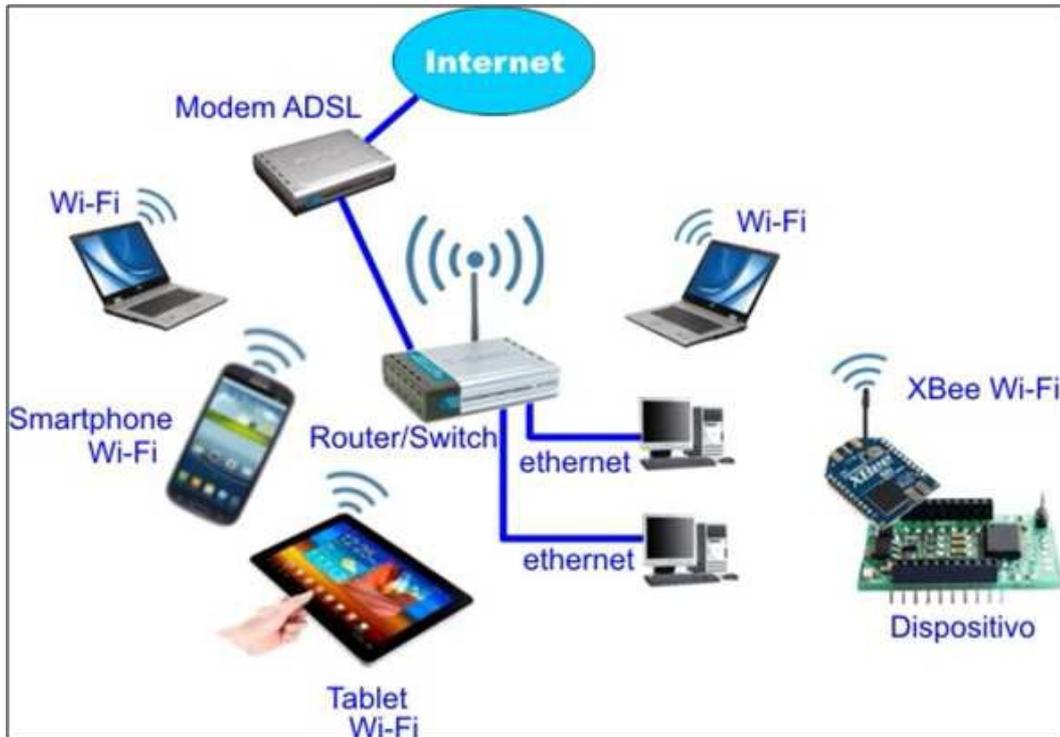
Os avanços na tecnologia de comunicação sem fio promoveram, paralelamente, o desenvolvimento de sistemas móveis na área das redes locais, ocasionando as redes locais sem fio denominadas de WLANs, as quais conseguem atingir uma taxa de transmissão mais alta, se comparado à rede de telefonia 3G (BRESIL, 2004). As WLANs conseguem atingir uma taxa de até 54 Mbps nos padrões 802.11a e 802.11g, e estudos apontam uma taxa de 100 Mbps com o padrão 802.11n. Mas, observa-se, na literatura, que o padrão predominante é o 802.11b, denominada como Wi-Fi cuja taxa máxima é de 11 Mbps.

Os padrões da WLANs apresentados anteriormente, foram desenvolvidos pelo IEEE (*Institute of Electrical and Electronics Engineers*), constituído no ano de 1963 após uma fusão entre as organizações AIEE (*American Institute of Electrical Engineers*) e IRE (*Institute of Radio Engineers*). O IEEE é uma organização formada por engenheiros, cientistas e estudantes, cujo ponto forte é o desenvolvimento de padrões para computadores e indústrias eletrônicas, em particular os IEEE 802, padrões desenvolvidos para redes locais amplamente atentos à evolução da comunicação (MITCHELL, 2015).

O funcionamento de uma rede sem fio é muito similar à da telefonia celular, em que toda comunicação deve necessariamente passar por uma central de controle, mesmo que os dispositivos móveis estejam a uma distância muito próxima um do outro. A comunicação é feita através de uma estação de suporte à mobilidade. Um exemplo dessa tecnologia de comunicação são os *hotspots*. Trata-se de redes locais públicas disponibilizadas em hotéis, restaurantes, bibliotecas, aeroportos, entre outros locais que vêm se adaptando para dar um suporte tecnológico para seus clientes (CORRÊA et al., 2006).

Os avanços nas redes móveis permitiram o surgimento da rede *ad-hoc*, fronteira final em uma comunicação sem fio e fator chave desse tipo de paradigma, pois relaciona a comunicação móvel e suas evoluções, já que a rede *ad-hoc* não precisa de uma infraestrutura para se comunicar como a rede local sem fio. Ou seja, ela permite que os nodos se comuniquem diretamente entre si ou através de múltiplos saltos dentro da rede, usando os receptores e transmissores sem fio. Essa é a diferença da rede *ad-hoc* para outras redes de comunicação tradicional. Na Figura 4, é possível constatar a estrutura da rede *ad-hoc*.

Figura 4 - Infraestrutura de uma rede *ad-hoc*.



Fonte: (MESSIAS, 2015).

A tecnologia da rede *ad-hoc*, é bem aceita, dadas suas diversas vantagens, como facilidade de aplicação; não depende de ponto que determine sua organização e controle; evita que o desempenho da rede seja afetado, caso algum nodo venha a falhar.

Com todas essas vantagens, muitas aplicações estão sendo realizadas com a rede *ad-hoc*, a exemplo da aplicação militar, redes de sensores, assistência a desastres, as WPAN (*Wireless Personal Area Network*), veículos autônomos não tripulados e as conexões com a internet, em razão do avanço da comunicação em rede na atualidade (CONTI; GIORDANO, 2014).

## 2.7 SISTEMAS OPERACIONAIS PARA DISPOSITIVOS MÓVEIS

Com tecnologia de processamento cada vez mais veloz e com mais memória de armazenamento, celulares, *smartphones* e dispositivos móveis ocupam cada vez mais tempo e espaço na vida das pessoas, pois, o que antes era feito somente pelos PCs, agora pode ser feito em qualquer lugar e hora do dia com um *smartphone*. Procurando garantir espaço na guerra da mobilidade com sistemas operacionais próprios, empresas como Nokia, Apple, Microsoft, Samsung, Google, Intel, entre outras, investiram pesado no setor (KAUR; SHARMA, 2014).

Dentre os muitos sistemas operacionais, este trabalho relata resumidamente os mais utilizados na evolução dos sistemas operacionais para celulares, como se pode observar na Tabela 2.

Tabela 2 - Sistemas operacionais móveis.

SO	Características	Aparelhos Celulares
Symbian OS	Iniciou em 1991, em 1997, foi lançado o Psion Serie5, primeiro <i>handhelds</i> a utilizar o sistema operacional EPOC 32 bits. Em 1998, houve a junção da Nokia, Motorola, Ericsson e a própria Psion, fundando assim, a companhia Symbian Ltd e posteriormente foi lançado o Ericsson R380, primeiro <i>smartphone</i> executando pelo Symbian OS. Em 2008, a Nokia comprou o Symbian Ltd, tornando-se a principal contribuinte para o código do Symbian OS, além de se consolidar no mercado com várias versões, utilizando novas tecnologias e ambientes gráficos inovadores (REZA; MAZUMDER, 2012).	Aparelhos que executam o Symbian OS: Lenovo P930, Motorola A1000, Nokia 3230, Sony Ericsson M600, BenQ P30, entre outros que suportam a tecnologia Symbian.
MeeGo	Surge em 2010, da junção do Maemo da Nokia e Moblin da Intel, baseados em Linux, cuja abrangência iria além dos <i>smartphones</i> , das televisões digitais, <i>desktops</i> , <i>tablets</i> , sistemas de entretenimento de automóvel e notebook. Depois de alguns problemas, foi descontinuado e ressurgiu em uma nova versão criada pela empresa Filandesa Jolla, agora conhecido como Sailfish OS (CARVALHO, 2012).	Vários dispositivos móveis eram executados com o sistema operacional MeeGo
Bada	Em 2009, a Samsung lançou, em Londres e Reino Unido a sua plataforma denominada Samsung Bada e sua SDK ( <i>Software Development Kit</i> ), juntamente com os benefícios que a sua tecnologia traria para seus parceiros, desenvolvedores de sistemas e consumidores da marca, mas sem o intuito de concorrer com as grandes companhias. Em 2013, parou com o Bada OS, mas, no ano seguinte a empresa lançou <i>smartphone</i> com o sistema operacional Tizen, para concorrer com o Galaxy S5 e o iOS da Apple, além de reformular seus relógios inteligentes, ou seja, uma das novidades da segunda geração do Galaxy Gear é a troca do sistema operacional Android para o Tizen (REUTERS, 2014)	Os <i>smartphones</i> executados com o Bada, foram classificados como Wave, a saber: Wave S8500, Wave 525, Wave 575, Wave 723, Wave Y, Wave M e o Wave 3. Sendo, o último da linha Wave.
BlackBerry OS-RIM	Criado pela empresa RIM ( <i>Research In Motion</i> ), em 1984 liderado por Mike Lazaridis. A partir de 2001, os dispositivos BlackBerry ficaram conhecidos como <i>smartphones</i> e, com o passar dos anos, foram disponibilizadas novas versões do sistema operacional e aparelhos mais sofisticados (MOREIRA, 2014).	O RIM 850 foi o primeiro, seguida pelos BlackBerry Q20 e BlackBerry Z3 executados pelo BlackBerry OS.

Fonte: Elaborado pelo autor.

Tabela 2 - Sistemas operacionais móveis (Continuação).

SO	Características	Aparelhos celulares
HP WebOS	<p>Desenvolvido em 1992 pela Palm Computing, por Jeff Hawkins, Donna Dubinski e Ed Colligan. Em 1993, foi lançado o ZOOMER PDA, comprado pela HP por ser baseado no núcleo do Linux com tecnologia Web destinadas a várias linguagens de programação, objetivando entrar no mercado dos <i>smartphones</i> e <i>tablets</i>. Após algumas atualizações do sistema e dispositivos, os mesmos não obtiveram os resultados satisfatórios, fato que levou a HP, em 2013, a anunciar a venda do WebOS a LG <i>Electronics</i>.</p> <p>Em 2014, a LG lança a sua <i>SmartTV</i>, executada com o sistema operacional WebOS (CALIL, 2012).</p>	Em 2011, foram apresentados dois telefones, o Veer e o Pre3 e um <i>tablet</i> executados pelo WebOS.

Fonte: Elaborado pelo autor.

Na Tabela 3 apresenta-se os sistemas operacionais móveis mais utilizados no mundo, disponibilizando uma tecnologia cada dia mais sofisticada para o usuário corporativo e convencional (KAUR; SHARMA, 2014).

Tabela 3 - Sistema operacional móvel mais utilizado na atualidade.

SO	Característica	Aparelhos celulares
Windows Mobile	<p>Desenvolvido pela Microsoft para executar dispositivos móveis como: <i>pocket</i>, <i>smartphones</i> e aparelhos de multimídia, em geral utilizando aplicações bem conhecidas no mundo dos PCs, como: o Word, Excel, Power Point, Windows Media Player, Internet Explorer.</p> <p>Em 2010 a Microsoft lança o Windows Phone 7 em dez dispositivos, entre esses a HTC, Dell, Samsung e LG, provando, assim, que poderia inovar no mercado dos <i>smartphones</i>.</p> <p>Em 2014, a Microsoft apresentou o Windows Phone 8.1, tendo como a principal novidade a compatibilidade com todas as versões do <i>smartphone</i> Lumia (MACIEJEWSKY, 2011).</p>	São várias marcas e modelos de <i>smartphones</i> executado pelo Windows Phone, a exemplo a linha Lumia.
iOS	<p>A Apple lançou em 2007, com o iPhone, o sistema operacional baseado no OS X, denominado de iPhone OS, tendo como base o UNIX específico para a linha de dispositivo móvel, voltado à simplicidade, beleza e eficiência de suas aplicações e a SDK (kit de desenvolvimento de software), que daria suporte ao desenvolvimento das aplicações, utilizando a linguagem de programação Objective C nativa da plataforma iOS (ABREU, 2012).</p> <p>Entre lançamento e atualizações das versões, já se vão oito anos e um mercado consolidado no ramo de dispositivos móveis, tanto que em 2015, a Apple apresentou a versão iOS 9. De acordo com a Apple (2015) “esta versão torna o fundamento do iOS ainda mais forte”, incluindo a otimização da bateria e a disponibilidades para a maiorias de seus dispositivos.</p>	A compatibilidade do sistema iOS com os dispositivos da linha iPhones, iPads e iPods Touch da Apple.

Fonte: Elaborado pelo autor.

Tabela 3 - Sistema operacional móvel mais utilizado na atualidade (Continuação).

SO	Característica	Aparelhos celulares
Android	<p>Desenvolvido pela empresa Android Inc., fundada em outubro de 2003, na cidade de Palo Alto na Califórnia por Andy Rubinera, Nick Sears e Chris White cujo objetivo era desenvolver um sistema operacional avançado para câmeras digitais, mas, ao longo da trajetória, direcionou seu desenvolvimento para os <i>smartphones</i>, porém, com poucos recursos e investimentos, se viu impossibilitada de competir com os sistemas operacionais conceituados na época (KAUR; SHARMA, 2014).</p> <p>Em agosto de 2005, a Google compra a Android Inc., colocando em prática seu desejo de trabalhar no mercado de dispositivos móveis. Liderado por Andy Rubine, desenvolveram uma plataforma móvel baseada em Linux com o objetivo de ser uma plataforma flexível, aberta e de fácil migração para os fabricantes de <i>smartphones</i> (QUERINO, 2013).</p> <p>Em 2007, o projeto intitulado como Android, construído sobre o <i>Kernel</i> do Linux, versão 2.6, foi lançado em parceria com a <i>Open Handset Alliance</i>, consórcio de empresas de tecnologias compostas por empresas como: Google, Sony, Samsung, operadoras de telefonia e fabricantes de dispositivos móveis liderada pela Google. O resultado desse projeto foi a construção da plataforma Android e o HTC Dream (G1), primeiro dispositivo móvel executado pelo SO Android, lançado em 2008 (LECHETA, 2013).</p> <p>A primeira versão do sistema foi o Android 1.0 Astro, as demais receberam o nome de sobremesas, como demonstra Guimarães (2013): <i>Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean</i> exceto a <i>Kit-Kat</i>.</p> <p>Com todas as versões e atualizações ocorrida ao logo de quase oito anos, o sistema operacional encontra-se na última versão denominada de Android Lollipop, lançada no final de 2014 e, em 2015, recebeu as devidas atualizações pela empresa Google (ANDROID, 2015a).</p>	<p>O sistema operacional Android é utilizado, desde <i>smartphones</i> e <i>tablets</i> a relógios, <i>Tvs</i>, carros. Ou seja, em variados dispositivos englobando variadas marcas e modelos.</p>

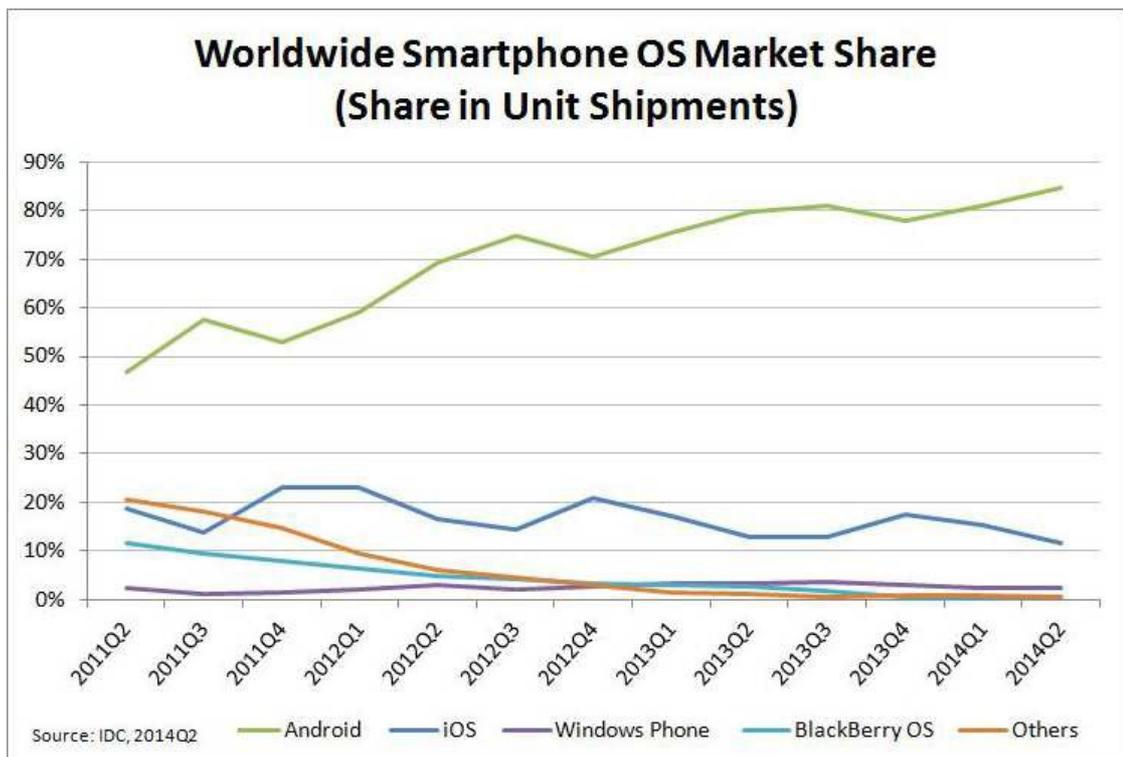
Fonte: Elaborado pelo autor.

## 2.8 CONSIDERAÇÕES FINAIS DO CAPÍTULO

A tecnologia digital está, como se viu, cada vez mais presente no cotidiano das pessoas, o que tem levado as empresas do setor a realizarem grandes investimentos na área. Os avanços tecnológicos nesse setor provocaram a criação das subáreas da computação móvel, redes de comunicação e as implementações de sistemas em dispositivos móveis, trazendo o que há de melhor em tecnologia computacional e digital, através das grandes empresas, como Nokia, Apple, Microsoft, Samsung, Intel, Google, entre muitas outras.

Hoje, existem *smartphones* de várias marcas e modelos disponíveis no mercado com vários tipos de sistemas operacionais. Tais aparelhos proporcionam ao usuário a realização de diversas tarefas do dia a dia na palma da mão. Na Figura 5, apresentam-se os sistemas operacionais mais utilizados na atualidade em dispositivos móveis.

Figura 5 - Os sistemas operacionais mais utilizados na atualidade.



Fonte: (HAMANN, 2014).

Observa-se, no gráfico apresentado na Figura 5, que o Android continua sendo líder absoluto no mercado móvel, provavelmente por ser o sistema mais disseminado no mundo não só de *smartphones*, mas de variados dispositivos executados pelo sistema operacional, e possuir a relação custo/benefício mais acessível às variadas classes da sociedade, dados os investimentos realizados pela Google e seus aliados em sua plataforma Android, possibilitando, desse modo, a execução de sua tecnologia tanto em aparelhos mais simples até os mais sofisticados.

No capítulo seguinte apresenta-se o cenário da tecnologia móvel, abordando o desenvolvimento de aplicações móveis relacionadas à plataforma de desenvolvimento Android e iOS, quanto a suas características, suportes fornecidos para o desenvolvimento e uma abordagem quanto ao custo/benefício, com o objetivo de expor para o leitor um panorama entre as duas plataformas mais utilizadas no mundo.

### 3 CENÁRIO DA TECNOLOGIA MÓVEL ABORDADA NO DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS

Nas últimas décadas, diversas empresas, em variados setores, vêm incorporando aplicações móveis na sua rotina de trabalho com intuito de agilizar seus negócios, integrando as aplicações com seu sistema de *back-end*, visando a lucros e mais lucros, já que os celulares e *smartphones* conectados à internet sincronizam informações diretamente de um servidor confiável da empresa na execução de aplicativos pertencentes a mesma.

Os avanços proporcionados pela tecnologia móvel, entretanto, criam dois mundos completamente distintos. Por um lado, têm-se as empresas e os desenvolvedores que buscam uma plataforma moderna e ágil para o desenvolvimento de aplicativos corporativos no auxílio de seus negócios e lucros; por outro lado, há os usuários convencionais que buscam um celular com visual elegante e moderno, fácil navegação e uma infinidade de recursos (LECHETA, 2013).

Diante da necessidade de agradar os dois segmentos, surgiram, ao longo dos anos, empresas especializadas em desenvolvimento de aplicativos móveis com plataformas específicas e tecnologias próprias. Entre as várias plataformas de desenvolvimentos de aplicativos móveis disponibilizados no mercado, temos duas, a saber, que detêm mais de 91% do mercado: a plataforma Android e a iOS (ZAPATA et al., 2014).

Sincronizado com a literatura vigente, abordou-se no presente trabalho o desenvolvimento de aplicações móveis, analisando a tecnologia e características próprias pertencentes a cada plataforma. Para tanto, realizou-se uma reflexão detalhada em vários artigos conceituados no meio acadêmico científico, constantes nos sistemas de busca ACM Digital.Library, IEEE Explorer, Google Scholar, periódicos CAPES/MEC e alguns livros relevantes na área de desenvolvimento de aplicativo móvel, que tomam as plataformas Android e iOS como ponto de interesse.

A partir das pesquisas bibliográficas e leituras realizadas em artigos selecionados como suporte teórico para o desenvolvimento do projeto proposto, destaca-se Mascarenhas e outros (2013), o qual apresenta um levantamento de vários trabalhos abordando o desenvolvimento de aplicações móveis, a saber: Gordrich e Rogers (2011), análise comparativa entre as plataformas móveis Android e iOS, a fim de verificar qual se adaptava melhor ao contexto acadêmico de ensino na sala de aula; Tarcy (2012), experiência do desenvolvimento de uma ferramenta utilizando as plataformas Android e iOS, analisando particularidades de interface gráfica em cada uma delas. Já Ribeiro et al. (2011), mostram comparações no desenvolvimento de aplicações Android e iOS, observados detalhes de cada plataforma no

contexto histórico, arquitetura, componentes fundamentais, interface com o usuário, recursos e muitas outras análises levantadas no decorrer do trabalho (MASCARENHAS et al., 2013).

Os trabalhos supracitados são alguns entre os que dão suporte a esta pesquisa e que demonstram a eficácia da tecnologia móvel, quando bem trabalhada, como é o caso do projeto Ubibus desenvolvido por Vieira et al. (2012b), cujo foco é tornar o sistema de transporte público mais inteligente, dinâmico e eficiente com auxílio de aplicativos que apresentam, para o usuário de transporte público, informações relevantes, como tempo de chegada prevista do ônibus e/ou fatores que afetam o trânsito, no sentido de auxiliar o usuário na tomada de decisões em sua locomoção (VIEIRA et al., 2012b). Nesse mesmo contexto, há o subprojeto do Ubibus: o Ubibus-Cars. Trata-se de um aplicativo desenvolvido na plataforma Android e iOS e tem como objetivo ser um sistema de caronas solidárias que visa a apoiar e a incentivar os indivíduos que compartilham de um mesmo trajeto a utilizar o mesmo veículo para chegar até o destino ou uma localidade próxima do seu objetivo de parada (VIEIRA et al., 2012a).

Segundo Mascarenhas et al. (2013), uma questão comum na concepção de uma aplicação móvel é a escolha da plataforma. Observa-se que, entre os critérios mais comuns na hora de escolher a plataforma, tem que se pensar no público-alvo; afinidade dos desenvolvedores com a plataforma, ferramenta e linguagem; o custo da licença de hardware para o desenvolvimento, ou seja, fatores consideráveis que independem da área, seja industrial, acadêmica, saúde, lazer etc.

No que diz respeito, ainda, à plataforma de desenvolvimento para dispositivos móveis, a literatura tem apontado duas plataformas em evidência no mercado de tecnologia móvel e digital: Android da Google e iOS da Apple (LECHETA, 2013).

### 3.1 DISTINÇÃO ENTRE AS PLATAFORMAS iOS E ANDROID

De acordo com a literatura vigente, vários fatores contribuíram e contribuem para o crescimento da tecnologia móvel no mundo. Nos tópicos anteriores, observaram-se algumas características das plataformas mais recorrentes, em especial no que diz respeito ao ambiente de desenvolvimento, facilidade de uso de ferramentas (hardware e software), suporte dado pelas respectivas comunidades de desenvolvimento, os softwares usados para simulação das aplicações, custos para desenvolvimentos nas plataformas, aspectos de distribuição de aplicações nos meios disponibilizados pelas plataformas (RIBEIRO; FERRAZ, 2011). Tal constatação objetivou mostrar como se constituem as plataformas Android e iOS, quanto à

natureza de suas tecnologias, ferramentas, bem como a relação custo/benefício de ambas, informações úteis no desenvolvimento do estudo/projeto proposto.

### 3.2 AMBIENTE DE DESENVOLVIMENTO

Na Tabela 4, demonstra-se que a plataforma Android provê maior flexibilidade quanto ao desenvolvimento de aplicações.

Tabela 4 - Ambiente de desenvolvimento.

<b>Itens</b>	<b>Android</b>	<b>iOS</b>
Multiplataforma (SO)	Windows, Linux, Mac Os	Mac Os
Multi-IDE	Eclipse, NetBeans, etc.	XCode
Linguagem Livre	Java, XML	Objective-C, Swift

Fonte: Elaboração do autor.

Logo, pode-se notar que o Android faz uso de várias plataformas no desenvolvimento de aplicativos móveis, a saber: Windows, Linux e Mac OS, enquanto que o iOS faz uso somente do Mac OS. Quanto a IDE, os aplicativos Android pode ser implementado em mais de uma, o iOS só é possível através da ferramenta XCode, enquanto a linguagem de programação utilizada em ambas estão sob licença livre.

### 3.3 COMUNIDADE DE DESENVOLVIMENTO

A plataforma Android possui uma grande comunidade de desenvolvimento, com fóruns e listas de discussões, além do *site* Android Developers disponibilizado pela Google com extensas documentações, sem falar de livros abordando o desenvolvimento de aplicações disponíveis no mercado. Com menor abrangência, a plataforma iOS também possui uma documentação bem estruturada com guias e exemplos para facilitar o trabalho dos desenvolvedores, além de documentações disponibilizadas no site iOS Dev Center, inclusive livros que abordam o desenvolvimento de aplicações, utilizando a plataforma iOS, o que leva a concluir que ambas possuem documentações gratuitas, códigos de exemplos de aplicações e fóruns de discussões para auxiliar o desenvolvedor em possíveis dúvidas.

### 3.4 SIMULAÇÃO DAS APLICAÇÕES

Para realizar os testes das aplicações desenvolvidas nas plataformas Android ou iOS,

ambas disponibilizam em seu núcleo uma ferramenta que possibilita ao desenvolvedor fazer ele mesmo os testes.

A SDK do Android disponibiliza para teste o Emulador e a XCode da plataforma iOS disponibiliza o Simulador, ambos com a mesma finalidade, qual seja: simular um dispositivo móvel com intuito de deixar a aplicação mais próxima possível da perfeição, como se tivesse rodando em um dispositivo real.

Quanto aos itens de cada ferramenta, apresenta-se um comparativo de 0 a 10 para medir o desempenho de cada grandeza, demonstrada na Tabela 5 (MASCARENHAS et al., 2013).

Tabela 5 - Simulação das aplicações.

<b>Itens</b>	<b>Android</b>	<b>iOS</b>
Desempenho	7	10
Suporte a sensores	8	10
Facilidade de uso	8	10
Experiência de usuário	7	10

Fonte: Elaboração do autor.

Dos resultados apresentados na Tabela 5, pode-se observar que tanto o desempenho, suporte a sensores, facilidade de uso e experiência de usuário se sobressaíram na iOS, o que talvez se explique pelo fato da Apple ser a fabricante do hardware e do software, situação bem diferente no Android, já que vários são os seus fabricantes de hardware.

### 3.5 CUSTO DE DESENVOLVIMENTO E DISTRIBUIÇÃO DE APLICAÇÕES ANDROID E iOS

Pensando a relação custo/benefício e as duas principais plataformas do mercado, a plataforma Android, cujo propósito é ser *open source*, flexível e inovadora, possui desenvolvimento mais acessível no mercado, enquanto o mesmo não acontece com sua principal concorrente, a plataforma iOS.

Na Tabela 6, apresentam-se alguns itens que compõem o custo do desenvolvimento e distribuição de aplicações para ambas.

Tabela 6 - Custo para desenvolvimento e distribuição de aplicações.

<b>Itens</b>	<b>Android</b>	<b>iOS</b>
Computador (mínimo valor)	R\$ 2.000,00	R\$ 4.000,00
Dispositivo (mínimo valor)	R\$ 400,00	R\$1.000,00
Fabricantes de dispositivos	mais de 50	1
Tipos de dispositivos	mais de 1.500	10
Canal de distribuição oficial	Google Play	Apple Store
Distribuição <i>standalone</i>	Sim	Não
Licença de desenvolvimento	\$ 25	\$ 99,00 ao ano

Fonte: Elaboração do autor.

Como se nota na Tabela 6, o Android possibilita um mercado mais acessível no quesito tecnologia mais barata e compatível com a confiabilidade da iOS.

### 3.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Para corroborar este estudo, bem como ratificar sua relevância, pesquisas foram realizadas em sistemas de buscas respeitáveis no meio acadêmico científico, livros conceituados na área e trabalhos de pesquisas que versam sobre o desenvolvimento de aplicações móveis. Em síntese, buscou-se contextualizar as características do desenvolvimento de aplicações utilizando a plataforma Android da Google e a plataforma iOS da Apple, com intuito de apresentar, para o leitor, as diferenças existentes entre as plataformas, além de oferecer-lhe um panorama quanto ao custo/benefício em ambas.

O capítulo seguinte apresenta a plataforma de desenvolvimento Google Android quanto a suas características e ferramentas disponibilizadas em seu núcleo, para auxiliar no desenvolvimento de aplicações para dispositivos móveis com o que há de mais moderno no mercado.

## 4 PLATAFORMA DE DESENVOLVIMENTO DE APLICAÇÕES PARA DÍSPPOSITIVOS MÓVEIS: GOOGLE ANDROID

Segundo a literatura, o Android foi uma grande aposta da empresa Google, juntamente com a OHA (*Open Handset Alliance*), para garantir boa parte do mercado voltado à tecnologia móvel. Com fortes investimentos, o Android vem ganhando fãs em diversos setores por se tratar de uma plataforma leve e extremamente customizável. Hoje, pode-se dizer que ele não se limita apenas a smartphones ou tablets, pois é possível encontrar aplicações Android, disponibilizadas pela Google, gerenciando vários tipos de dispositivos, como televisores, micro-ondas, geladeiras, carros e relógios (ANDROID, 2015a).

Pelos investimentos proporcionados pela Google e a OHA, o Android se tornou, nos últimos anos, líder no mercado dos sistemas operacionais móveis, com mais de 800 mil aplicativos disponibilizado na Google Play Store, alcançando a marca de 1 milhão no ano 2013, o que comprova sua excelente aceitação, e, claro, a marca de 75% do mercado desse tipo de dispositivo (QUERINO, 2013).

### 4.1 FERRAMENTAS QUE CONSTITUEM A PLATAFORMA GOOGLE ANDROID

A plataforma Google Android disponibiliza para o desenvolvedor uma variação de recursos que facilita sua vida na hora de programar uma determinada aplicação. As ferramentas que fazem parte desta tecnologia são: Android SDK, o IDE Eclipse com *plugin* ADT (*Android Development Tools*) e a linguagem de programação Java. A escolha dessas ferramentas se deve ao fato da equipe de desenvolvedores ter maior familiaridade com elas e por escolher fazer uma aplicação nativa (MASCARENHAS et al., 2013).

#### 4.1.1 Android SDK

O Android SDK é um *kit* de desenvolvimento de software lançado no mercado pela Google, no ano de 2008, para desenvolver aplicações para Android. Possui em sua base ferramentas e bibliotecas necessárias para construir e testar aplicações específicas para o Android. Entre as ferramentas disponibilizadas pela SDK, constam um emulador para simular o celular, ferramentas utilitárias e uma API (correspondente ao código que identifica a API *level*, específica em cada versão da plataforma Android) completa para a linguagem Java, com todas as classes necessárias para desenvolver as aplicações (LECHETA, 2013). Na Figura 6, é apresentada a página de download do Android SDK.

Figura 6 - *Download do Android SDK.*

**Get the Android SDK**

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android.

If you're a new Android developer, we recommend you download the ADT Bundle to quickly start developing apps. It includes the essential Android SDK components and a version of the Eclipse IDE with built-in ADT (Android Developer Tools) to streamline your Android app development.

With a single download, the ADT Bundle includes everything you need to begin developing apps:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- The latest Android platform
- The latest Android system image for the emulator

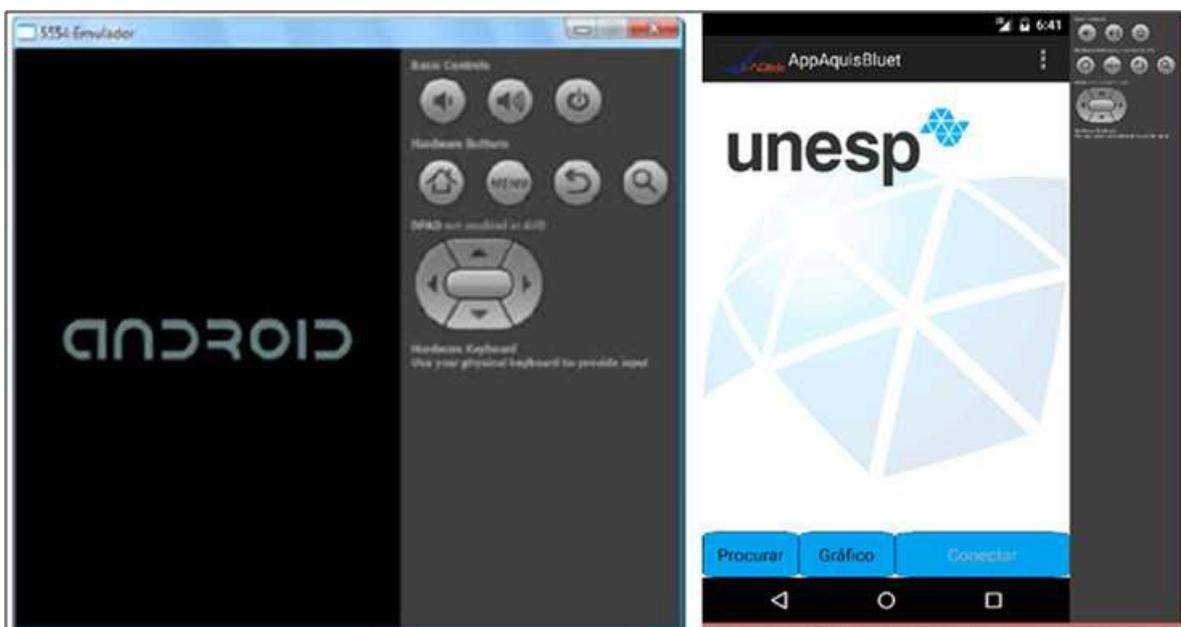
**Download the SDK**  
ADT Bundle for Windows

Fonte: (SILVA, 2013).

#### **4.1.1.1 Emulador**

O Emulador é uma ferramenta utilizada no Eclipse, baseada em uma aplicação de código aberto, chamado QEMU, que tem por função emular vários tipos de hardwares. Representa o funcionamento de um celular Android, no sentido de poder executar no emulador, quase tudo, que poderá ser executado no Android, uma vez que a sua maior vantagem é proporcionar uma melhor consistência ao software desenvolvido, para que o mesmo possa funcionar perfeitamente em aparelhos que executam o Android no mercado atual (PEREIRA; SILVA, 2009). Na Figura 7, apresenta-se o emulador disponibilizado para executar as aplicações Android.

Figura 7 - Emulador do Android executando o AppAquisBluet.



Fonte: Adaptado de (SILVA, 2013).

#### 4.1.1.2 Principais APIs

Pereira e Silva (2009) relatam que as principais APIs disponibilizadas no Android têm a função de facilitar a criação de aplicações, além de permitir a criação de uma interface com o usuário, criação de telas, acessar arquivos, criptografar dados, ou seja, utilizar todas as funcionalidades definidas pelo usuário das APIs em questão.

De acordo com Querino (2013), para usufruir da tecnologia disponibilizada pela SDK, o desenvolvedor terá que obter sistemas operacionais específicos que execute o Android SDK. Entre esses sistemas têm-se:

- Windows XP (32-bit) Vista (32 ou 64-bits) ou Windows 7, 8, 10 (32 ou 64-bits);
- Mac OS X 10.5.8 ou posterior (somente X86);
- Linux (testado no Linux Ubuntu).

Além dos sistemas específicos, precisa-se de um ambiente de desenvolvimento que suporte o SDK, como o Eclipse 3.6.2 (Helios) ou superior, utilizando o *plug-in ADT* para integrar o emulador ao Eclipse e a JDK (*Java Development Kit*) (LECHETA, 2013).

### 4.1.2 JDK (*Java Development Kit*)

O JDK oferece ferramentas como o compilador Java, utilizados por IDEs e SDKs para desenvolvimento de programas em Java e, para ter uma interação com o Eclipse, o JDK disponibiliza em seu núcleo um JRE (*Java Runtime Environment*), responsável por permitir a execução de programas Java com o Eclipse (MEDNIEKS; DORNIN; NAKAMURA, 2012).

### 4.1.3 Eclipse

Segundo Mednieks, Dornin e Nakamura (2012), o Eclipse é uma plataforma de propósito geral, que, além de ser aplicada na criação de IDEs para diferentes linguagens de programação, também pode ser utilizada na criação de IDEs personalizados para muitas SDKs específicas.

Geralmente, o Eclipse é utilizado como uma IDE capaz de escrever, testar e depurar software, especialmente em linguagem Java. Encontram-se, na literatura, várias IDEs e SDKs derivadas do Eclipse para diversos tipos de software Java (MEDNIEKS; DORNIN; NAKAMURA, 2012). Na Figura 8, apresenta-se a primeira página do Eclipse, depois da instalação.

Figura 8 - Primeira tela do Eclipse.



Fonte: (MEDNIEKS; DORNIN; NAKAMURA, 2012).

#### 4.1.4 *Android Developer Tools (ADT)*

De acordo com a literatura pesquisada, é possível desenvolver uma aplicação para o Android, utilizando a linguagem de programação Java, tanto no Eclipse, Netbeans, IntelliJ IDEA ou Android Studio, como em ambiente de desenvolvimento. O Google teve a preferência pelo Eclipse, por esse permitir a instalação do *plug-in* ADT, que disponibiliza funções que facilitam o desenvolvimento, os testes e a compilação do projeto desenvolvido.

A vantagem em utilizar o ADT se deve ao fato que possibilita executar o emulador do Android diretamente do Eclipse, além de ser possível controlá-lo, visualizando *logs* e simulando o envio de uma mensagem SMS, ligação telefônica. Outrossim, ainda é possível o envio de arquivos para o emulador, entre outras funções, diretamente do Eclipse com o *plug-in* ADT (LECHETA, 2013).

Segundo Querino (2013), além do SDK do Android, o Eclipse e o *plug-in* ADT do Eclipse têm-se disponibilizado para *download* o *ADT Bundle*, um Eclipse contendo todas as configurações estabelecidas e prontas para serem usadas no desenvolvimento de aplicações Android.

## 4.2 ARQUITETURA DO ANDROID

A plataforma Android é uma tecnologia móvel completa, que envolve pacotes com programas para celulares, constituído por sistema operacional, *middleware* (um programa de computador que faz a mediação entre software e demais aplicações), aplicativos e interface do usuário, de modo que, por ser uma plataforma *open source*, pode ser sempre adaptada a fim de incorporar novas tecnologias, conforme forem surgindo no mercado.

Diante das tecnologias disponibilizadas pela Google em sua plataforma Android, pode-se observar como está estruturada a arquitetura Android utilizada no desenvolvimento de aplicações (SMITH; FRIESEN, 2012).

Silva (2013) diz que a estrutura da plataforma Android é constituída por aplicações, *framework*, bibliotecas, *Android runtime* e *Linux kernel*, cada uma com sua funcionalidade e características próprias. Na Figura 9, seguindo a sequência apresentada por Silva (2013), podem-se observar as camadas que compõem a plataforma Android desenvolvidas pela Google e OHA.

Figura 9 - Arquitetura geral da plataforma Android.



Fonte: (SILVA, 2013).

#### 4.2.1 Aplicações

Aplicações correspondem à camada no topo da estrutura do Android, onde se encontram todos os aplicativos fundamentais escritos em Java para o Android. Entre esses aplicativos, têm-se: um cliente de e-mail, mapas, navegadores, calendários, programas de SMS, gerenciador de contatos, agendas, entre outros. À medida em que forem sendo desenvolvidos pela comunidade Android, serão colocados no topo da estrutura (PEREIRA; SILVA, 2009).

#### 4.2.2 Application Framework

De acordo com Pereira e Silva (2009), na camada de *Application Framework*, encontram-se todas as APIs (*Application Programming Interface*) e os recursos utilizados pelos aplicativos. Dentre os elementos presente nesta camada, os autores relatam:

- **Gerenciador de Atividade** (*Activity Manager*) - tem a função de gerenciar o ciclo de vida de todas as *activities*, quando iniciá-las e quando terminá-las, possibilitando, assim, o deslocamento de uma *activity* para outra e assim sucessivamente;

- **Gerenciador de Pacote** (*Package Manager*) - é um meio de comunicação com o sistema para dizer quais os pacotes estão sendo utilizados no dispositivo e quais são as capacidades destes pacotes, função esta utilizada pelo *activity manager* para ler as informações dos APK's (pacotes de arquivos do Android);
- **Gerenciador de Janela** (*Windows Manager*) - é o componente responsável pelo gerenciamento das apresentações das janelas, informando qual janela estará ativa e assim sucessivamente;
- **Sistema de Visualização** (*View System*) - tem a função de disponibilizar todo o tratamento gráfico para a aplicação, como botões, *layouts* e *frames*;
- **Provedores de Conteúdo** (*Content Providers*) - têm a função de encapsular dados (como os *bookmarks* uma linha livre de marcadores e um serviço de armazenamento, do *App Browser*) que podem ser compartilhados entre aplicativos;
- **Gerenciador de Localização** (*Location Manager*) - permite que um dispositivo Android esteja ciente de sua localização física;
- **Gerenciador de Notificação** (*Notification Manager*) - tem a função de permitir que um aplicativo notifique o usuário de um evento significativo (como a chegada de uma mensagem) sem interromper o que o usuário está fazendo;
- **Gerenciador de Recurso** (*Resource Manager*) - este componente tem a função de permitir que um aplicativo acesse seus recursos;
- **Gerenciador do Telefone** (*Telephony Manager*) - permite que um *App* (aplicativo) aprenda sobre os serviços de telefonia de um dispositivo.

Observou-se que todos os componentes disponíveis na plataforma Android possuem pleno acesso às APIs utilizadas pelo núcleo da plataforma, no sentido de que qualquer aplicação pode publicar as suas capacidades e qualquer outra pode fazer uso desta capacidade, desde que obedeça às restrições de segurança aplicada pelo quadro (PEREIRA; SILVA, 2009).

### 4.2.3 Bibliotecas

A camada de bibliotecas carrega consigo um conjunto de bibliotecas C / C++ utilizadas pelo sistema, as quais podem ser acessadas por *frameworks* disponibilizados para o desenvolvedor. Dentre essas bibliotecas Pereira e Silva (2009) destacam:

- ***Freetype*** - suporta *bitmap* (formato de imagem, antigo e ocupa muito espaço) e renderização de fontes vetoriais;

- **Libc** - é uma implementação derivada da biblioteca padrão do BSD (*Berkeley Software Distribution*) do C, sintonizada com dispositivos baseados em Linux;
- **LibWebCore** - corresponde a um *web browser engine* (software que compõe páginas HTML) utilizado tanto no *Android browser* quanto para exibições *web*;
- **Mídia Framework** - baseadas em *PacKetVideo OpenCORE*, suportam a reprodução e geração de muitos formatos populares de áudio e vídeos e trabalham com arquivos de imagens estáticas, suportando formatos MPEG4, H.264, MP3, AAC, AMR, JPEG e PNG;
- **OpenGL|ES** - biblioteca responsável pelos gráficos 3D que fornecem uma aplicação baseada na APIs OpenGL|ES 1.0, além de utilizar aceleração de hardware e software 3D;
- **SGL** - responsável pelo fornecimento da implementação gráfica 2D;
- **SQLite** - responsável por fornecer um eficiente e leve mecanismo de banco de dados relacional, disponível para todos os aplicativos. Implementado em C, leve e embutida, com suporte a base de dados acima de 2 TB;
- **SSL** - fornece *sockets* (é um ponto final de um processo de interfluxo de comunicação através de uma rede de computadores) seguros e baseados em camadas de segurança (*SSL-based*) para comunicação de rede;
- **Surface Manager** - responsável por fornecer os subsistemas de exibição, como as camadas de aplicações 2D e 3D.

#### 4.2.4 *Android Runtime*

A camada de *Android Runtime* é formada pela *Dalvik Virtual Machine* e pelo *Core Libraries*, sendo que a *Dalvik* uma máquina virtual com melhor desempenho, maior integração com a nova geração de hardware, projetada para executar várias máquinas virtuais paralelamente. Além disso, funciona em sistemas com baixa frequência de CPU, pouca memória RAM disponível em sistema operacional sem espaço de *Swap* (gerenciamento de memória). Outra característica da *Dalvik*, é executar arquivos DEX (*Dalvik Executable*) classes de Java convertidas para a máquina virtual através da ferramenta DX, constituída juntamente com o SDK do Android. A *Dalvik VM* baseia-se no *kernel* do Linux para funcionalidades subjacentes como o encadeamento e gestão de baixo nível de memória, além de

ser otimizada para consumo mínimo de memória, bateria e CPU.

Pereira e Silva (2009) definem que o *Core Libraries* inclui um grupo de bibliotecas que fornece a maioria das funcionalidades disponíveis nas principais bibliotecas da linguagem de programação Java, como estruturas de dados, acesso a arquivos, acesso a redes e gráficos.

#### 4.2.5 *Linux Kernel*

O *Linux Kernel* (versão 2.6), é a última camada que constitui a plataforma Android. Oferece serviços básicos paralelos, como: *threading* (linha de controle), gerenciamento de memória de baixo nível, pilha de rede, gerenciamento de processos e um modelo de *drives*. Além dessas funções, age como uma camada de abstração entre o hardware e o resto da pilha do software. Outra característica relevante nesta camada é o fato dela ser um poderoso sistema próprio de gerenciamento de energia, no qual os *drives* de energia do *Kernel* controlam a energia utilizada pelos dispositivos em suas aplicações. As aplicações que não estão sendo utilizadas pelo aplicativo são desligadas para que se tenha uma economia de energia (PEREIRA; SILVA, 2009).

### 4.3 CICLO DE VIDA DA *ACTIVITY* NO ANDROID

Segundo Lecheta (2013), a *activity* é umas das classes mais importantes no Android, pois é responsável por controlar os eventos da tela e definir qual *View* será responsável por desenhar a interface gráfica do usuário. Normalmente, a *activity* representa uma única tela em sua aplicação, podendo ser uma visualização, criação ou edição de dados, pois, na maioria dos aplicativos Android, consta diversas *activities* (PEREIRA; SILVA, 2009).

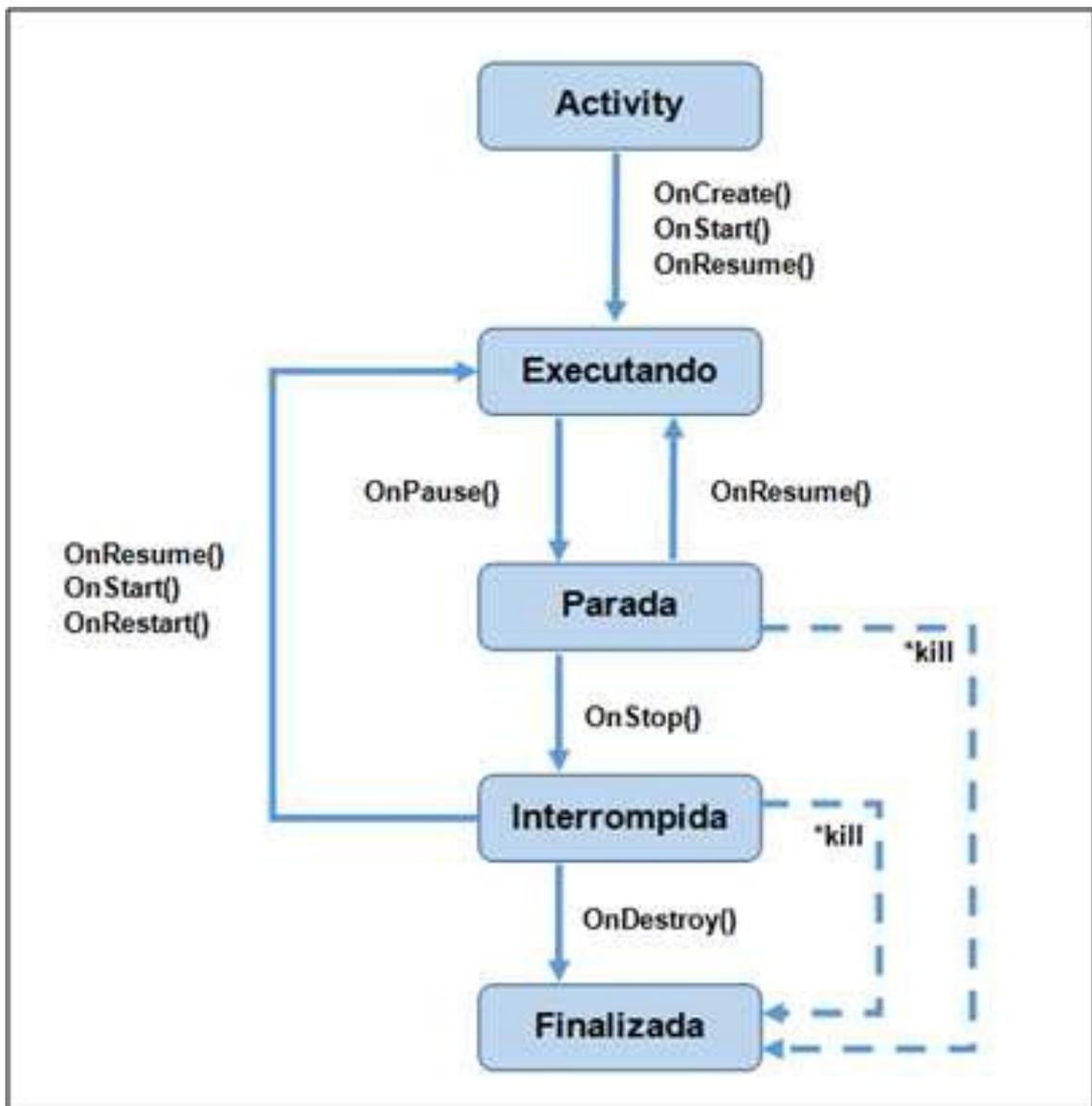
Os eventos de uma classe *activity* podem ser utilizados para controlar o estado da aplicação, sendo necessário a implementação do evento *onCreate*, método da *activity* responsável pela inicialização e execução da aplicação, tendo em vista que os eventos compõem o ciclo de vida de uma *activity* nos seguintes estados, mencionados por Lacheta (2013):

- **Executando** - a atividade que estiver ativa no visor do dispositivo;
- **Parada** - uma atividade que perdeu o foco para outra atividade, mantém todas as informações de estado, porém não está interagindo com o usuário, podendo ser finalizada em situação de baixo nível de memória disponível;
- **Interrompida** - caso uma atividade não esteja sendo utilizada pelo usuário, ela mantém as suas informações de estados. Porém, são muitas vezes finalizadas quando uma recuperação de memória for necessária, perdendo as informações;

- **Finalizada** - o sistema pode remover uma atividade da memória, caso esta esteja interrompida temporariamente ou se estiver parada. O estado anterior só pode ser restaurado se os métodos tiverem sido implementados pelo desenvolvedor.

Na Figura 10, pode-se observar o ciclo de vida da *activity*, conforme pontuado por Pereira e Silva (2012):

Figura 10 - Ciclo de vida da *Activity* do Android.



Fonte: Adaptado de (PEREIRA; SILVA, 2012).

- **onCreate** - método chamado quando a atividade é inicialmente criada;
- **onStart** - chamada quando a atividade torna-se visível para o usuário;
- **onResume** - é o topo da pilha de atividade, chamado quando vai iniciar a interação com o usuário;

- ***onRestart*** - chamado quando sua atividade estiver interrompida e prestes a ser acionada pelo usuário;
- ***onPause*** - chamado quando o sistema está perto de começar uma próxima atividade.

Geralmente é utilizado para gravar as informações que ainda não foram salvas;

- ***onStop*** - é chamado quando a atividade não estiver mais sendo utilizada pelo usuário e tenha perdido o foco para outra atividade;
- ***onDestroy*** - pode ser chamado quando a atividade terminou ou quando o sistema precisa finalizar a atividade para liberação de recursos;
- ***onFreeze*** - possibilita salvar o estado atual da atividade, quando a atividade está preste a ser suspensa.

#### 4.4 CARACTERÍSTICAS PRÓPRIAS DA PLATAFORMA ANDROID

Existem várias plataformas destinadas ao desenvolvimento de aplicações móveis, as quais disponibilizam para o desenvolvedor um ambiente repleto de inovações tecnológicas com características próprias. Diante dessas inovações, Pereira e Silva (2012) demonstram as características principais constituídas na plataforma Android:

- *Framework* de aplicação que emite a incorporação, a reutilização e a substituição de recursos de componentes;
- Máquina Virtual Dalvik, uma máquina virtual otimizada para dispositivos móveis;
- Gráfico otimizado por meio de uma biblioteca 2D e gráfico 3D baseado na especificação *Open-GL 1.0* (aceleração de hardware é opcional);
- SQLite, um gerenciador de banco de dados para armazenamento de dados estruturados;
- Suporte multimídia para formatos de som (MP3, AAC, AMR), vídeo (MPEG4 E H.264) e formatos de imagens (JPG, PNG, GIP) nativamente;
- Possibilita mais integração com as tecnologias GSM, mas apresenta limitações dependentes do hardware;
- Suporta a tecnologia *bluetooth* EPGE, 3G e Wi-Fi, porém possui as mesmas limitações de hardware dos recursos de telefonia GSM;
- Câmeras, GPS, bússola e acelerômetro que permite integração, mas ambos os casos dependem das limitações do hardware;
- Poderoso ambiente de desenvolvimentos incluindo um emulador de dispositivo, ferramenta para depuração, analisador de memória e desempenho.

#### 4.4.1 Java

A linguagem de programação Java é utilizada para programar aplicações na plataforma Android, disponibilizando ao usuário todos os seus recursos. Normalmente, por se tratar de uma linguagem de programação orientada a objeto, que ao ser compilada gera um *bytecode* que representa um conjunto de instruções que será executado em uma máquina virtual Java (*JVM - Java Virtual Machine*), se tornou a linguagem principal da plataforma Android (MEDNIEKS; DORNIN; NAKAMURA, 2012).

#### 4.4.2 Google Play

A Google Play, antes conhecida como *Android Market*, foi criada para auxiliar a distribuição das aplicações do Android, além de ser uma meio de divulgação da plataforma Android criada pela Google e a OHA. O objetivo da loja virtual é fornecer aos desenvolvedores de aplicativos um lugar comum para disponibilizar suas aplicações para serem baixadas por usuários. Para utilizar a loja virtual, o desenvolvedor precisa pagar uma taxa de US\$ 25,00 e concordar com os termos de uso. Nesse processo, o desenvolvedor tem 70% dos lucros dos aplicativos vendidos nela (LECHETA, 2013).

#### 4.4.3 Código aberto e livre

Por ser uma plataforma de código aberto (*open-source*), a plataforma Android evolui de forma mais constante, já que programadores de diversos países contribuem para o melhoramento dela. Outro fator que contribui para sua evolução é a utilização do sistema operacional por fabricantes de celulares sem necessidade de pagar pela utilização do sistema. Além disso, o Android possui a licença *Apache Software Foundation (ASF)*, que permite fazer alterações no código fonte, criando produtos customizados, sem precisar compartilhar as alterações (LECHETA, 2013).

### 4.5 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo, apresentou-se a plataforma Android criada pela empresa Google, juntamente com um grupo de investidores de vários seguimentos da telefonia móvel, formando, assim, a aliança OHA, liderada pela Google.

A OHA, com intuito de revolucionar o mercado de aplicações móveis no mundo, disponibilizou para o desenvolvedor, corporativo ou convencional, uma plataforma moderna, flexível e *open-source*, baseada no *kernel* do Linux, além de disponibilizar em seu núcleo uma SDK repleta de ferramentas para auxiliar o desenvolvedor na criação de aplicações

cada vez mais sofisticadas. Para usufruir de toda essa tecnologia, o desenvolvedor pode fazer o *download* da *SDK ADT Bundle*, instalar seguindo tutoriais na internet ou em livros, e começar a desenvolver aplicações que poderão figurar na loja virtual Google Play Store para serem baixadas pelos usuários, desde que dentro das normas estabelecidas pela Google.

O capítulo a seguir demonstra como foi desenvolvido o sistema de aquisição de dados via *bluetooth* e quais ferramentas foram utilizadas para o desenvolvimento e realizações de testes, para que se chegasse em um produto final: o aplicativo AppAquisBluet.

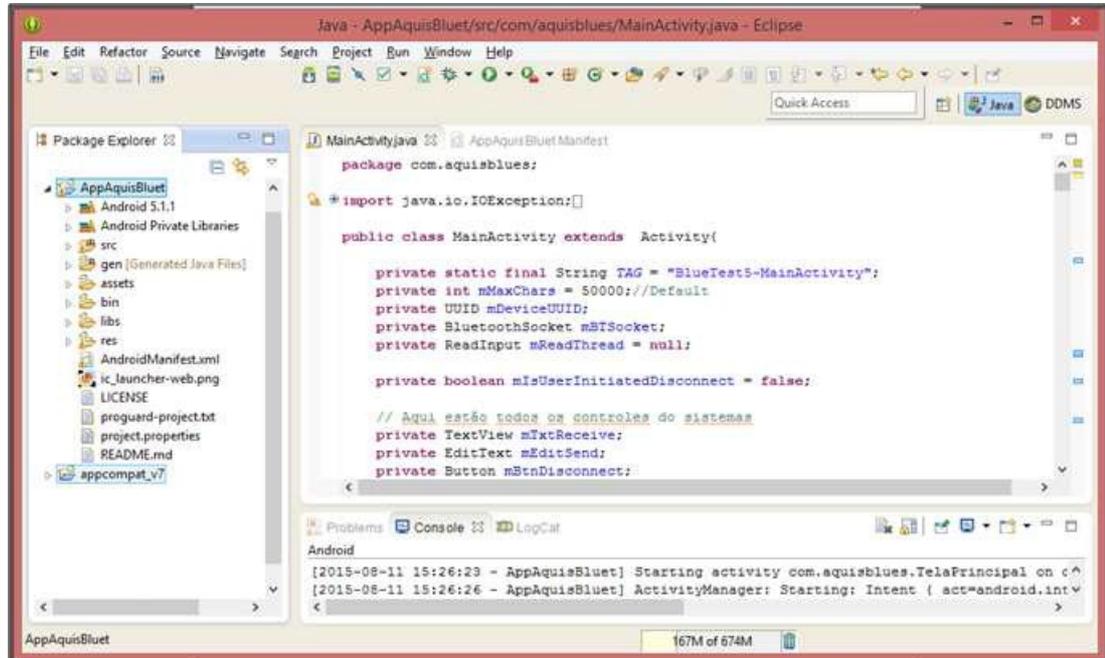
## 5 MÉTODOS UTILIZADOS NO DESENVOLVIMENTO DO APPAQUISBLUET

Neste capítulo, aborda-se a estrutura de desenvolvimento do aplicativo AppAquisBluet a partir da plataforma Android, utilizando, para tal, da tecnologia e ferramentas disponibilizadas pela Google e seus aliados. O foco é a estrutura do projeto na IDE Eclipse *ADT Bundle*, o arquivo *AndroidManifest.xml*, *Activities* e *Intents*, componentes da tela do AppAquisBluet e a navegação nas telas.

### 5.1 ESTRUTURA DO PROJETO APPAQUISBLUET NA IDE ECLIPSE

Observa-se, na criação de um projeto Android, utilizando a IDE Eclipse, como é o caso do AppAquisBluet, que seu núcleo é constituído de várias pastas, arquivos e bibliotecas. Para que o projeto funcione perfeitamente, alguns elementos são essenciais, conforme demonstrado na Figura 11, isto é, a IDE Eclipse composta pelas principais pastas do projeto em destaque.

Figura 11 - Estrutura do projeto na IDE



Fonte: Elaborada pelo autor.

Como se observa na aba de *Packager Explorer* da IDE, têm-se as principais pastas que compõem o projeto AppAquisBluet, responsáveis pelo funcionamento do aplicativo, e definidas por Lecheta (2013), como:

- **src** - a pasta que contém todos os arquivos fontes em java do projeto, como a classe *MainActivity* que foi criada pelo *wizard*;
- **gen** - contém todos os arquivos gerados automaticamente pela IDE. Nesta pasta encontra-se o arquivo responsável pela interface entre a codificação Java e XML, chamado *R.java*. Este arquivo é gerado pelo compilador que referencia todos os recursos do projeto, e o mesmo jamais deve ser alterado pelo desenvolvedor;
- **Android 5.1.1** - contém o arquivo *android.jar*, que possui todas as classes necessárias para uma aplicação Android;
- **assets** - contém arquivos HTML, arquivos texto, banco de dados, entre outros;
- **bin** - onde fica armazenado o arquivo de saída final do projeto, como o arquivo APK;
- **libs** - pasta que armazena as bibliotecas de terceiros, que estão em uso no projeto;
- **res** - possui algumas subpastas como: *layout*, *menus*, *values*, *drawable* e XML com configurações variadas, e serão automaticamente geradas na classe R, dependendo do conteúdo;
- **AndroidManifest.xml** - arquivo que descreve a especificação de uma aplicação e que contém expressamente cada um de seus componentes, além de ser responsável por definir permissões, bibliotecas, versão do sistema e componentes de software que a aplicação necessitar;
- **project.properties** - arquivo de configuração gerado automaticamente, contém informações como a versão do pacote a ser utilizado para compilação.

A pasta *AndroidManifest.xml* é a base de uma aplicação Android, é obrigatória e deve ficar sempre na pasta raiz do projeto em desenvolvimento, uma vez que receberá toda a configuração necessária para executar aplicação. Ela armazenará o nome do pacote e a versão do SDK utilizado no projeto, as aplicações, as *activities* que devem ser executadas, *intent-filter* para filtrar e executar a *activity*, permissões para usar alguma biblioteca específica e várias outras configurações que serão armazenadas, dependendo do tipo de projeto (LECHETA, 2013).

### 5.1.1 Activity

Além das pastas que formam a estrutura de um projeto, a *activity* também faz parte do processo, pois corresponde a uma das classes mais importantes no Android, por ser responsável por controlar os eventos da tela e definir qual *View* desenhará a interface gráfica do usuário implementando o método *onCreate (bundle)* e executando a aplicação chamando o método *set-ContentView (view)*, já que esta *activity* é obrigatoriamente declarada no *AndroidManifest.xml* para que a aplicação funcione perfeitamente.

Quanto ao ciclo de vida de uma *activity*, observam-se os possíveis estados em que esta pode se encontrar, ou seja: executando, temporariamente interrompida, em segundo plano ou completamente destruída. Na seção 4.3, relatou-se o ciclo de vida de uma *activity*, ratificando tal ciclo na Figura 13 do tópico 5.2.

### 5.1.2 Intent

Segundo Lecheta (2013), a classe *android.content.Intent* é considerada uma das partes fundamentais do Android, pois está presente em todos os lugares e representa uma mensagem da aplicação para o sistema operacional, solicitando que algo seja realizado, além de seu importante papel na arquitetura do Android na integração de diferentes aplicações.

A *Intent* é definida em português como a **intenção** que uma determinada aplicação tem em realizar uma tarefa. Entre essas intenções, podem-se apresentar: enviar uma mensagem para o sistema operacional, abrir uma nova tela da aplicação utilizando o método *startActivity(intent)*, solicitar ligações, abrir *browser*, abrir o *Google Play* entre outras. Na Figura 12, pode-se observar a classe *intent* sendo utilizada no projeto AppAquisBluet, no *AndroidManifest.xml*, declarando nome e categoria. Além disso, se outras *activity* quiserem chamar esta nova, devem fazer pelo nome do *intent filter*.

Figura 12 - Activity e Intent-filter implementado no *AndroidManifest.xml* do projeto.

```
<activity
    android:name="unesp.appaquisbluet.Homescreen"
    android:configChanges="keyboardHidden|orientation"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Fonte: Elaborada pelo autor.

### 5.1.3 Broadcast Receiver

A classe *Broadcast Receiver* é bem semelhante a uma *Activity*, mas sem interface de usuário própria, passiva de informações externas provenientes de outras aplicações. Ela se mantém em um estado inativo até que um evento, ao qual deve estar previamente programado, seja executado em segundo plano, disparando, assim, uma série de atividades descritas pelo programador em um determinado projeto (MEDNIEKS; DORNIN; NAKAMURA, 2012).

### 5.1.4 Service

A classe *Service* do Android é utilizada para executar um processamento em segundo plano, por tempo indeterminado, sem conter qualquer interface de interação com o usuário. Ou seja, é utilizada para executar tarefas independente do que está sendo exibido na tela do dispositivo, é equiparável às *Activities* quanto a sua capacidade de execução e funcionalidades em um projeto (LECHETA, 2013).

## 5.2 BLUETOOTH

A comunicação *bluetooth* iniciada em 1994, pela companhia *Ericsson*, utiliza tecnologia de sinais de radio de baixo custo, a fim de permitir a comunicação entre telefones celulares e acessórios, deixando de lado os tradicionais cabos. Criou o *MC-Link*, um sistema de rádio de curto alcance com uma implementação relativamente simples e barata. Logo, por ser uma tecnologia fácil e barata, despertou o interesse de outras empresas, tanto que, no ano de 1998, foi criado o consórcio *Bluetooth SIG (Bluetooth Special Interest Group)*, formado pelas companhias *Ericsson*, *Intel*, *IBM*, *Toshiba* e *Nokia* (dezenas de outras companhias aderiram ao consórcio com o passar do tempo) (BLUETOOTH, 2015). Esta diversidade de empresas focadas no projeto *bluetooth* foi importante para permitir o desenvolvimento de padrões que garantissem o uso e a interoperabilidade da tecnologia nos mais variados dispositivos. A partir de então, o *bluetooth* começou a virar realidade. Em julho de 1999, o grupo de trabalho IEEE 802.15 WPAN (*Wireless Personal Area Network*) propôs a especificação *bluetooth* versão 1.0 (THOMPSON; KUMAR; KLINE, 2008).

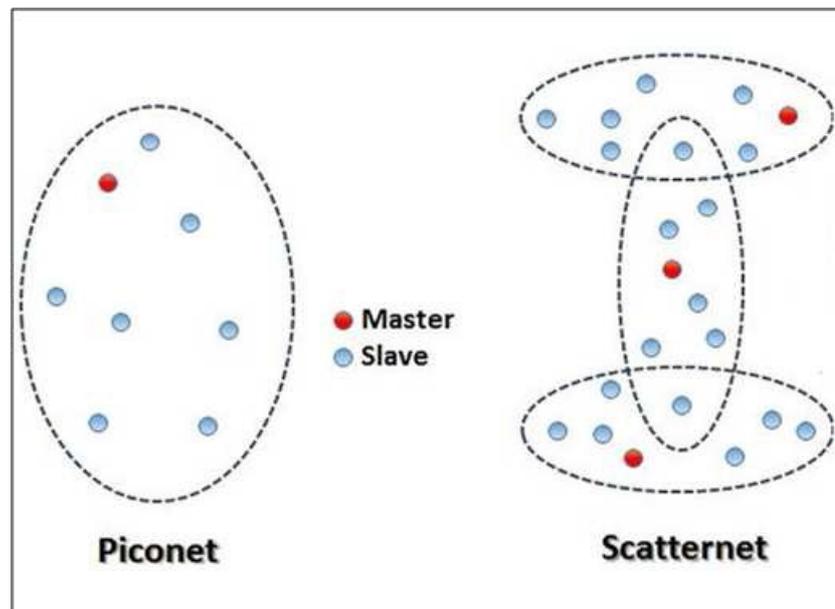
De acordo com Bluetooth (2015), com o passar dos anos, foram disponibilizada novas versões da tecnologia *bluetooth* com o intuito de melhorar a taxa de frequência na comunicação e diminuir o consumo da bateria. Atualmente, a mesma encontra-se disponível no mercado na versão 4.1, lançada em 03 de dezembro de 2013. Contudo, para que seja possível atender aos mais variados tipos de dispositivos, o alcance máximo do *bluetooth* foi dividido em três classes:

- Classe 1: potência máxima de 100 mW (*miliwatt*), alcance de até 100 metros;
- Classe 2: potência máxima de 2,5 mW, alcance de até 10 metros;
- Classe 3: potência máxima de 1 mW, alcance de até 1 metro.

É importante frisar, no entanto, que dispositivos de classes diferentes podem se comunicar sem qualquer problema, basta respeitar o limite daquele que possui um alcance menor. Diante do exposto, foi estabelecida a comunicação *bluetooth* do aplicativo AppAquisBluet para trabalhar na Classe 2 com a transmissão de dados a faixa de 24 Mb/s.

Quando a comunicação entre dois ou mais dispositivos é estabelecida utilizando uma conexão *bluetooth*, essa comunicação estabelece uma rede denominada *piconet*, originando, assim, duas nomenclaturas: o *master* (mestre) e o *slave* (escravos), contexto em que o *master* representa o dispositivo que abriu a conexão e tem como tarefa regular a transmissão de dados na rede e o sincronismo entre os outros dispositivos que serão denominado de escravos. A rede *piconet* pode suportar até 8 dispositivos, um *master* e 7 *slaves*. No entanto, é possível elevar este número a partir da sobreposição de *piconets*, denominada de *scatternet*. Com isso, uma *piconet* se comunica com outra que esteja dentro do limite de alcance. Na Figura 13, apresenta-se as *piconet* e *scatternet*, onde um dispositivo *slave* pode fazer parte de mais de uma *piconet* e o *master*, ocupando apenas uma posição em cada *piconet* (BLUETOOTH, 2015).

Figura 13 - Ilustração de *piconet* e *scatternet*.



Fonte: (THOMPSON; KUMAR; KLINE, 2008).

Para que haja a conexão em uma *piconet*, os dispositivos utilizam-se de um método de identificação para saber quais outros dispositivos fazem parte de sua *piconet*. Aqueles que desejam se conectar a uma *piconet* já existente podem emitir um sinal denominado de *Inquiry* e aqueles dispositivos que receberem o sinal irão responder com um pacote FHS (*Frequency Hopping Synchronization*), informando a sua identificação e os dados de sincronização da *piconet*. Com base nestas informações, o dispositivo pode, então, emitir um sinal chamado *Page*, para estabelecer uma conexão com outro dispositivo e fazerem as trocas de dados entre os periféricos (CHLAMTAC; CONTI; LIU, 2003).

Contudo, para que haja uma economia de energia, faz-se uso de um terceiro sinal denominado *Scan*, que tem como função fazer com que os dispositivos que estiverem ociosos entrem em *stand-by*, ou seja, operem em um “modo de descanso”, poupando energia (CHLAMTAC; CONTI; LIU, 2003).

A implementação das novas versões na tecnologia *bluetooth*, no decorrer dos anos, possibilitou a comunicação com outros dispositivos, propiciando ao usuário dessa tecnologia a possibilidade de presenciar uma grande quantidade de cenários de comunicação *bluetooth* possíveis.

Devido ao crescimento e a aceitação no mercado mundial, a tecnologia *bluetooth* vem apresentando um novo paradigma de comunicação, despertando o interesse das grandes empresas na adesão dessa tecnologia em seus produtos, o que não foi diferente para a Google, já que esta incluiu em sua plataforma Android o suporte para a pilha da rede *bluetooth*, que permite que um dispositivo troque dados sem necessidade de cabos com outros dispositivos *bluetooth*, desde que o *framework* da aplicação forneça acesso à funcionalidade *Bluetooth* através das APIs *bluetooth* do Android. Logo, as APIs permitem que a aplicação se conecte a outros dispositivos que tenha *bluetooth*, como: teclados, *mouses*, *tablets*, impressoras, caixas de som, fones de ouvidos, dispositivos móveis e pessoais, além de permitir a conexões entre PC e celulares. Ativam-se assim, os recursos de rede sem fio ponto a ponto e multiponto, liberando a comunicação dos periféricos que queiram se comunicar via *bluetooth*, respeitando a classe de alcance máximo de cada dispositivo (LECHETA, 2013).

### 5.3 API *BLUETOOTH* PARA ANDROID

Atualmente, as quatro principais tarefas necessárias para se realizar uma comunicação via *bluetooth* são: configurar o *bluetooth*, encontrar dispositivos que estejam emparelhados ou disponíveis na área, conectar os dispositivos e transferir dados entre os mesmo. Mas, para que essa ação seja concretizada, precisa-se iniciar um *link* de comunicação, utilizando os *sockets bluetooth*, tornando possível a ação de transmitir e receber dados entre os dispositivos.

Utilizando a API *bluetooth* do Android, o dispositivo *bluetooth* local é controlado através da classe *BluetoothAdapter*, que representa o dispositivo Android em que a aplicação está sendo executada. Dessa forma, para acessar o *adaptador bluetooth* padrão da API, é necessário utilizar o método *getDefaultAdapter()*. Contudo, para dar início a busca por dispositivos *bluetooth*, devem-se incluir as permissões do *Bluetooth* no *AndroidManifest.xml* e modificar as propriedades do dispositivo local, onde a permissão de *BLUETOOTH\_ADMIN* deve ser concedida (LECHETA, 2013).

Após concedida a permissão do *BLUETOOTH\_ADMIN* incluída no *AndroidManifest.xml*, é possível habilitar e desabilitar o *bluetooth* do dispositivo, usando os métodos *enable e disable*. A descrição das classes dessas API, utilizadas para configurar o *bluetooth*, procurar por dispositivos, conectar-se aos dispositivos pareados e transferir dados entre os dispositivos, é definida por Lecheta (2013) como:

- ***BluetoothAdapter*** - representa o adaptador *bluetooth*, em que, através dessa classe, é possível procurar por outros dispositivos e se conectar a eles;
- ***BluetoothDevice*** - representa o outro dispositivo *bluetooth*, que pode solicitar ao dispositivo remoto uma conexão ou informações do dispositivo;
- ***BluetoothSocket*** - representa a interface para um soquete *bluetooth*;
- ***BluetoothServerSocket*** - representa um servidor aberto para atender a requisições de outros dispositivos, função essa que funciona como um *host*.

Na Figura14, apresentam-se as linhas de códigos a serem adicionadas no arquivo *AndroidManifest.xml*, para que seja concedida a comunicação entre os dispositivos.

Figura 14 - Permissão *Bluetooth* no *AndroidManifest*.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Fonte: Elaborado pelo autor.

Segundo Lecheta (2013), o *adaptador bluetooth* oferece métodos para leitura e configuração das propriedades do hardware *bluetooth*, em que o adaptador somente poderá ler ou configurar, se estiver habilitado no dispositivo. Caso não esteja habilitado, o método *getDefaultAdapter()* retorna *null*. Este teste é feito no *onCreate()* da *activity* principal do projeto, para saber se o dispositivo está habilitado com esta função. Na Figura 15, tem-se o código para verificar se o *Bluetooth* está ativo no dispositivo.

Figura 15 - Código para verificar se o *bluetooth* está ativo.

```
mBTAdapter = BluetoothAdapter.getDefaultAdapter();
if (mBTAdapter == null) {
    Toast.makeText(getApplicationContext(), "Bluetooth não encontrado", Toast.LENGTH_SHORT).show();
}
```

Fonte: Elaborado pelo autor.

Um das ações corriqueiras por parte dos usuários é manter o *Bluetooth* desabilitado até utilizá-lo, para conservar a carga da bateria. Mas, para habilitar o adaptador *Bluetooth*, pode-se usar a constante *BluetoothAdapter.ACTION\_REQUEST\_ENABLE* com o método *startActivityForResult()*, contexto em que aparecerá uma tela pedindo que o usuário habilite o *Bluetooth*. Na Figura 16, apresenta-se o código utilizado para habilitar o *bluetooth* no dispositivo.

Figura 16 - Código de solicitação da ativação do *bluetooth*.

```
else if (!mBTAdapter.isEnabled()) {
    Intent enableBT = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBT, BT_ENABLE_REQUEST);
} else {
    new SearchDevices().execute();
}
```

Fonte:Elaborado pelo autor.

### 5.3.1 Tornando Visível o Dispositivo Remoto para Liberar a Comunicação com o Adaptador Bluetooth

Para que a comunicação via *bluetooth* seja possível, é necessário que o usuário habilite a função visível no dispositivo remoto, e que o adaptador *bluetooth* esteja configurado como visível, através do método *getScanMode* da classe *BluetoothAdapter*, uma vez que esse método retorna uma das seguintes constantes representada a seguir:

- *SCAN\_MODE\_CONNECTABLE\_DISCOVERABLE* - significa que o adaptador local *Bluetooth* pode ser descoberto por qualquer outro dispositivo;
- *SCAN\_MODE\_CONNECTABLE* - significa que este não está descoberto para novos dispositivos remotos, ou seja, só pode conectar-se a dispositivos que já o descobriram anteriormente;
- *SCAN\_MODE\_NONE* - nenhum dispositivo remoto pode encontrar o adaptador *bluetooth* local.

Observa-se, na API do *bluetooth* Android, que a permissão de habilitar o *bluetooth* tornando-o visível é uma questão de privacidade, já que, por padrão, ela fica desabilitada. Assim, somente uma permissão explícita do usuário, autorizando a visibilidade iniciada por meio de uma nova *Activity* usando a constante *ACTION\_REQUEST\_DISCOVERABLE*, é possível. Além disso, seguindo um padrão imposto pela API, o dispositivo local fica visível por dois minutos, podendo ser alterado este tempo com a constante *EXTRA\_DISCOVERABLE\_DURATION* na execução do *Intent*.

Para obter a confirmação do usuário quanto a permissão ou não para que o adaptador *bluetooth* fique apto a ser descoberto por outros dispositivos *bluetooth*, pode-se sobrescrever o *onActivityResult()* e verificar o parâmetro *resultCode*. Caso seja permitido, será indicado o tempo em que adaptador *bluetooth* local ficará visível para ser descoberto e, caso seja negada, indicará que o usuário rejeitou o pedido (ANDROID, 2015b).

Segundo Android (2015b), existem outros métodos úteis da classe *BluetoothAdapter* para encontrar dispositivos que estejam no limite das classes de transmissão de dados, como:

- *isDiscovering()* - a mesma retornará um *boolean* que indica se o dispositivo está ou não fazendo uma busca por outros dispositivos remotos;
- *startDiscovery()* - o método iniciará uma busca por dispositivos que se encontram dentro do alcance do adaptador *Bluetooth* local;
- *cancelDiscovery()* - neste momento será cancelada a procura por outros dispositivo.

### 5.3.2 Estabelecendo Canais RFCOMM na Comunicação *Bluetooth*

Segundo Bluetooth (2015), as API's do *bluetooth* para Android baseiam-se no protocolo de comunicação RFCOMM (*Radio Frequency Communications*), simulando um protocolo de substituição de cabo, usado para criar uma porta serial virtual, resultando num mecanismo para abrir soquetes de comunicação entre dois dispositivos *bluetooth* que estejam pareados. Para estabelecer essa comunicação *bluetooth* RFCOMM bidirecional, o programador pode utilizar as classes: *BluetoothServirSocket* e *BluetoothSocket*, em que a primeira será implementada em um dispositivo se passando por servidor que atende e aceita pedidos de conexão, e a segunda será implementada como um cliente. Após estabelecida a conexão entre os dispositivos, será feita a transferência de dados por intermédio da classe *BluetoothSocket*.

Tanto a classe *BluetoothServirSocket* como a *BluetoothSocket* utilizam-se do UUID (*Universally Unique Identifier*) para se comunicar, um formato padronizado para ID *string* de 128 bits, usado para identificar de forma única a informação. A vantagem de se utilizar o UUID é o fato dele ser grande o bastante, o que inibe colisões de identificadores.

O método *listenUsingRfcommWithServiceRecord(String, UUID)* é usado para deixar o dispositivo na **espera** por pedidos de conexão, de modo que os parâmetros de identificação do servidor e o identificador universal devem ser confirmados pelo cliente para que possa se conectar ao servidor. Cria-se, assim, um RFCOMM *BluetoothSocket*, pronto para iniciar uma conexão de saída segura para este dispositivo remoto, usando *lookup* SDP (*Service Discovery Protocol*), de UUID com método *createRfcommSocketToServiceRecord(UUID)*. Se

a conexão for concluída com sucesso, o método *accept()* retornará um *BluetoothSocket* e este pode ser usado para transferência de dados entre os dispositivos.

Depois de conectados, terá apenas um *BluetoothSocket* para o dispositivo cliente e o servidor. Contudo, pode-se afirmar que o conceito estabelecido entre o cliente e servidor só foi utilizado para exemplificar como funciona a comunicação entre os dispositivos até a conexão ser estabelecida. A partir deste momento, não existe distinção entre os dispositivos, pode-se enviar e receber dados através do *BluetoothSocket*, pois a transferência de dados entre os dispositivos é tratado pelas classe Java *OutputStream* e *InputStream*, obtidas do *BluetoothSocket*, através dos métodos *getOutputStream* e *getInputStream* (LECHETA, 2013).

#### 5.4 BIBLIOTECA GRAPHVIEW

Para apresentar resultados de dados, graficamente, no projeto AppAquisBluet, utilizou-se a *GraphView*, uma biblioteca para Android com objetivo de criar programaticamente diagramas flexíveis e de boa aparência, já que é fácil de entender, integrar e personalizá-los. Quanto às características da biblioteca disponibilizada para o Android, de acordo com *GraphView* (2015), tem-se na versão *GraphView-3.1.4.jar*:

- Dois tipos de gráfico, sendo o de linhas e o de barras;
- Desenha várias séries de dados, definindo cor e descrição para cada uma e que será apresentada mais de uma série em um gráfico;
- Mostrar legenda, podendo ser exibida na linha do gráfico, definindo a largura e alinhamento vertical (superior, médio, baixo);
- Etiquetas personalizadas, em que as etiquetas para o eixo-x e eixo-y são geradas automaticamente. Entretanto, pode-se definir suas próprias etiquetas;
- Lidar com dados incompletos, sendo possível mandar os dados em frequência diferente;
- Pode-se limitar a janela de exibição para que apenas uma parte dos dados seja exibida;
- Pode-se rolar os dados com um movimento de toque do dedo no visor do dispositivo, ou seja, escala de gestos;
- Desde a versão Android 2.3, podem ser usados dois dedos para tocar no visor, usando escala gesto (*Multi-touch*), assim, o visor pode ser alterado;
- *Background* (gráfico de linha). Opcionalmente desenha um fundo claro sob o novo diagrama;
- Aplica-se um manual de limites no eixo Y, ou seja, o desenvolvedor limita o eixo Y;
- Apresenta-se as configurações de dados em *Realtime Graph (Live)*.

Diante do exposto, é possível alterar os dados do gráfico em tempo de execução, ou seja, em tempo real. Para fazer isso, disponibilizam-se dois métodos importante nas subclasses da série:

- ***resetData*** - este método redefine todos os dados, para que os dados atuais sejam substituídos pelo novo;
- ***appendData*** - métodos que acrescentam um único conjunto com os dados atuais recebidos em uma nova aquisição. Há também a função ***scrollToEnd***, que irá percorrer a *GraphView* automaticamente para o último valor de X (GRAPVIEW, 2015).

## 5.5 BANCO DE DADOS

O banco de dados é definido como um sistema computacional que armazena informações por meio eletrônico, objetivando futuras consulta, alterações ou exclusão desses dados, caso seja necessário (MEDNIEKS; DORNIN; NAKAMURA, 2012). Segundo Paulo (2014), para tornar esses dados seguros e confiáveis, os desenvolvedores criam estruturas de dados complexas e formas de ocultá-las dos usuários, com objetivo de determinar os níveis de segurança através das camada de abstração de dados, como pode ser observado a seguir:

- **Nível físico** - descreve em detalhes as estruturas de dados que são armazenadas;
- **Nível lógico** - esse nível de abstração tem por finalidade realizar a definição dos dados que serão armazenados, bem como as relações existentes entre eles;
- **Nível *view*** - é o mais próximo do usuário e descreve apenas parte do banco de dados. Existe para simplificar a interação com o sistema, fornecendo diversas visões diferente para o mesmo banco de dados (PAULO, 2014).

### 5.5.1 Elementos Básicos de um Banco de Dados

A estrutura do banco de dados definida por Smith e Friesen (2012) é constituída por conceitos de tabelas, registros, chaves e índices.

- **Registro** - coleção de informações armazenadas sequencialmente em campos contextualizados, em que cada campo é considerado uma coluna;
- **Tabelas** - estruturas construídas através de uma aglomeração de diversas colunas, cada uma com seu nome próprio, para armazenar e recuperar futuros registros. Normalmente as tabelas são contextualizadas para receber informações com a mesma finalidade;
- **Chaves** - são especificadas para identificar uma linha completa de um determinado registro, podendo ser classificadas em chave primária, responsáveis pela identificação

de um registro; chave secundária, diferentes da chave primária, no sentido que podem ser repetidas e é habitualmente usada para dar uma melhor organização aos campos. E, por último, a chave estrangeira, campos que podem realizar a relação entre duas tabelas diferentes;

- **Índices** - fornecem um acesso mais rápido à um determinado registro em uma tabela, durante a pesquisa, eliminando a necessidade de realizar uma varredura completa da tabela em questão.

### 5.5.2 Modelo de Dados

O modelo de dado pode ser descrito através do projeto de um banco de dados, utilizando os níveis físico, lógico e *view*, fazendo uso de uma coleção de conceitos de descrição de dados, relações, semântica e restrições das informações. Date (2004) define esses modelos da seguinte forma:

- **Modelo relacional** - utiliza-se de tabelas, onde cada uma possui um nome e armazena registros e formato fixo de vários tipos diferentes. Com isso, cada tabela contém registros de um tipo específico de informações;
- **Modelo entidade relacionamento** - é estruturado através da criação de abstrações que representam a relação entre objetos conhecidos como entidades, podendo ser representadas por um objeto do mundo físico;
- **Modelo de dados baseado em objetos** - é uma extensão do modelo de entidade relacionamento, que utiliza conceitos de orientação a objeto para estender seus recursos;
- **Modelo de dados semiestruturado** - no modelo de dados semiestruturado, é possível que dados do mesmo tipo possam ter diferentes conjuntos de atributos. Com isso, podem ser utilizados por meio da linguagem XML.

### 5.5.3 Linguagem de banco de dados

Segundo Mednieks, Dornin e Nakamura (2012), a linguagem de banco de dados mais utilizada pelo modelo de dados relacional é a linguagem de consultas estruturada *SQL* (*Structured Query Language*). A sua popularidade e confiabilidade deve-se ao fato dela poder ser dividida em duas categorias, conforme se observa a seguir:

- **Linguagem de manipulação de dados** - conhecida pela sigla *DML* (*Data Manipulation Language*), é responsável pela manipulação das informações no banco de dados através de operações de inserção, exclusão, alteração e recuperação de dados;
- **Linguagem de definição de dados ou DDL** (*Data Definition Language*) - é a codificação que define o esquema do banco de dados, bem como a especificação e propriedades adicionais dos dados em questão.

#### 5.5.4 Banco de Dados *SQLite*

O sistema operacional Android contém um banco de dados nativo denominado *SQLite*, pelo qual são fornecidas bibliotecas para realizar operações de manipulação de dados em uma determinada aplicação (LECHETA, 2013).

O *SQLite* é definido como um banco de dados estável, de código aberto, amplamente utilizado em dispositivos com limitações no volume de memória ou armazenamento de dados. Seu diferencial consiste na ausência de um processo executado no sistema operacional exclusivo para o servidor. Isso acontece porque o *SQLite* escreve diretamente em arquivos comuns no sistema de armazenamento em disco (SQLITE, 2015).

Uma aplicação Android se conecta com o banco *SQLite* através de uma interface denominada *DbHelper*. Para criar tal interface, é necessário que se crie uma classe auxiliar que realize a conexão através da utilização de outra classe contida no *framework* do sistema Android, denominada de *SQLiteOpenHelper*, que retorna uma instância do tipo *SQLiteDatabase* (MEDNIEKS; DORNIN; NAKAMURA, 2012).

As operações básicas de inserção, alteração, remoção e execução de consultas personalizadas estão presentes como parte da plataforma. Elas podem ser acessadas através de funções contidas no *framework* do sistema operacional. As funções, quando executadas, retornam um conjunto de colunas para um ponteiro contendo as informações solicitadas no código, no qual é possível acessar uma por vez por meio de comandos determinado por meio de código (SMITH; FRIESEN, 2012).

### 5.6 MODELAGEM DA APLICAÇÃO APPAQUISBLUET

As funcionalidades do aplicativo serão apresentadas através da linguagem de modelagem UML (*Unified Modeling Language*), que fornece diagramas padronizados, que ajudam a descrever e projetar sistemas de software, voltada a aplicações orientadas a objetos, com objetivo de especificar, documentar e estruturar o desenvolvimento completo de um sistema de informação, seja ele móvel ou web (BEZERRA, 2015).

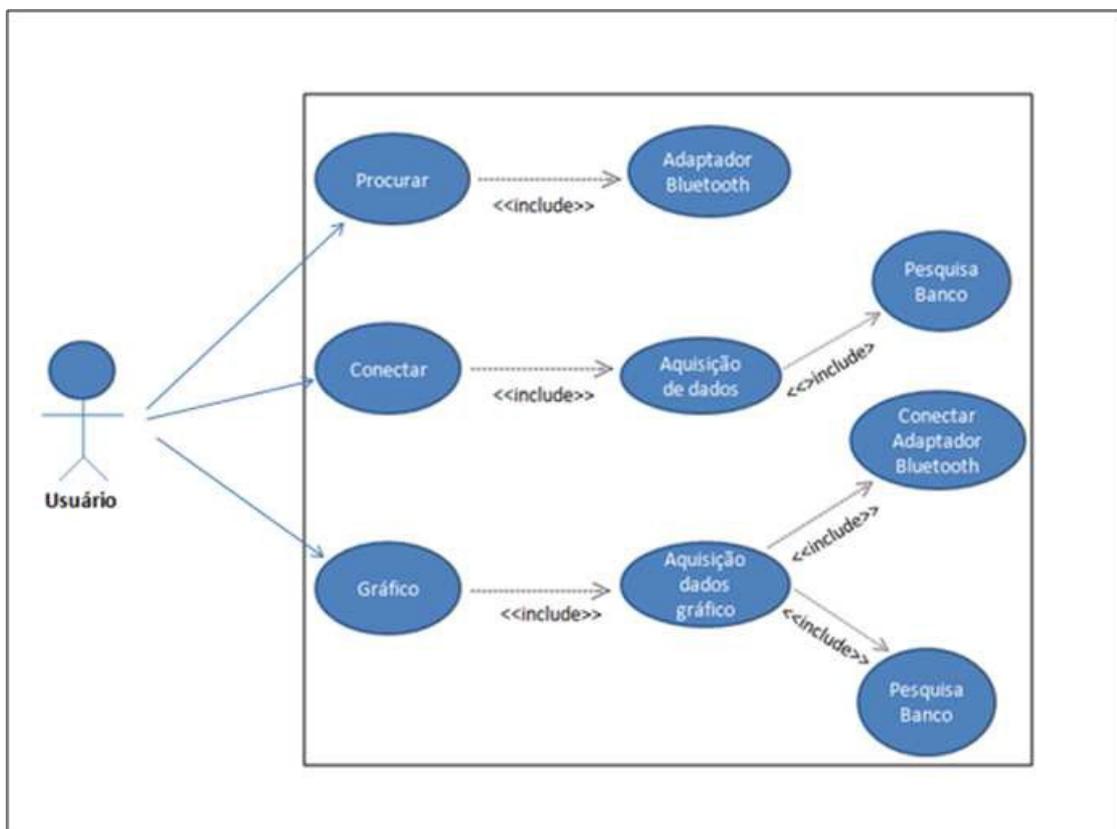
A UML possui diagramas, que são usados em combinação, com a finalidade de obter todas as visões e aspectos do sistema, ou seja, são representações gráficas do modelo parcial de um sistema em desenvolvimento (PARADA; BRISOLARA, 2012).

Nesta seção, demonstram-se os principais diagramas da modelagem criados para o desenvolvimento do aplicativo AppAquisBluet: o diagrama de caso de uso, diagrama de classes e diagrama de fluxo de telas. A criação desses diagramas foi efetuada com recurso ao plugin **ObjectAid UML Explorer** instalado no Eclipse, pois, através desta ferramenta, é possível efetuar *reverse engineer* ao código existente e gerar vários tipos de diagramas UML.

### 5.6.1 Diagrama de Caso de Uso

O diagrama de caso de uso do aplicativo desenvolvido neste trabalho indica as funcionalidades e as interações do usuário executadas na aplicação para obter aquisições de dados em tempo real, intermediada pela comunicação *bluetooth*. Na Figura 17, expõe-se o diagrama composto por desenhos simples, que descrevem de maneira bem objetiva o que textualmente poderia ficar bem extenso.

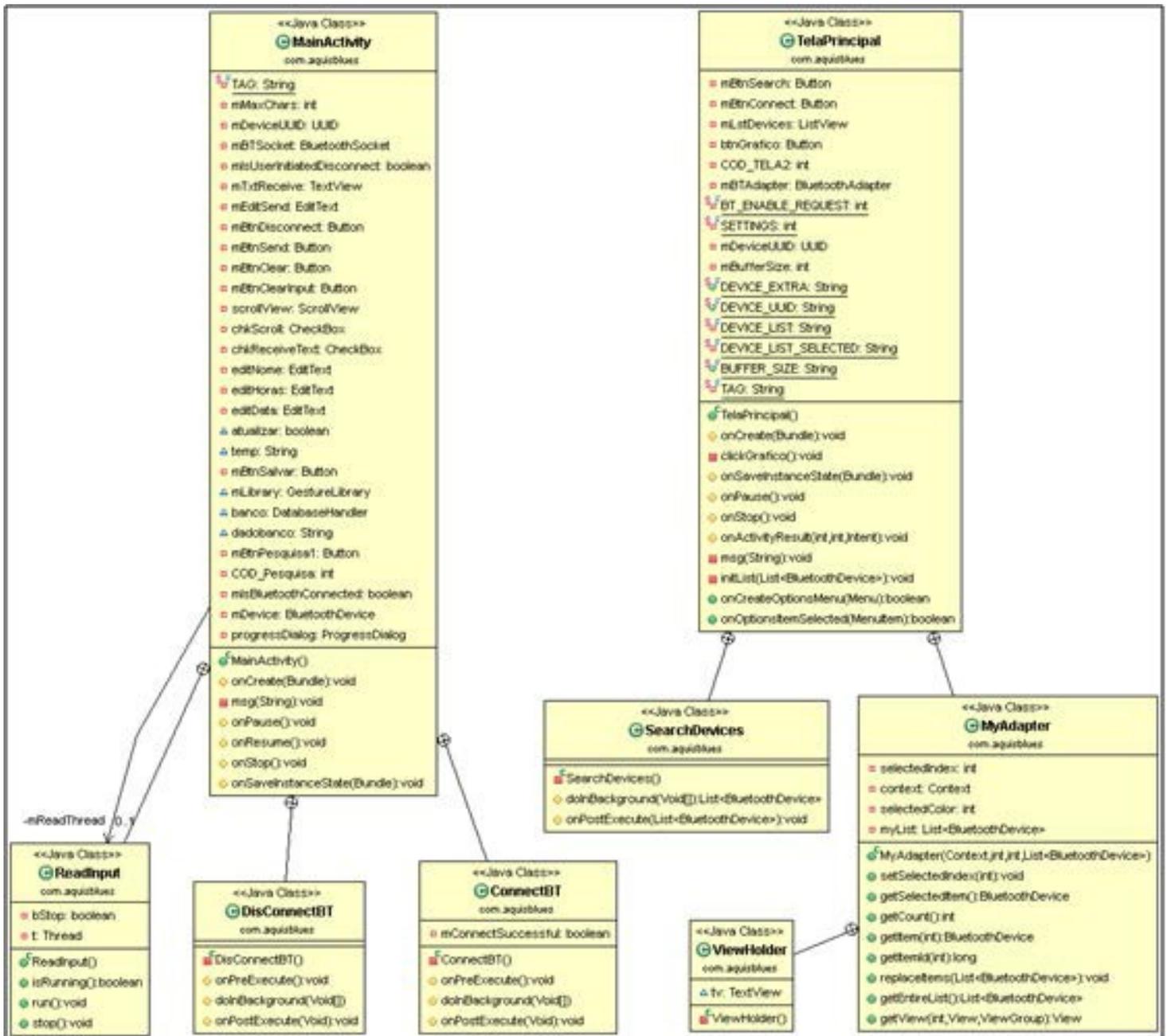
Figura 17 – Diagrama de caso de uso.



Fonte: Elaborado pelo autor.

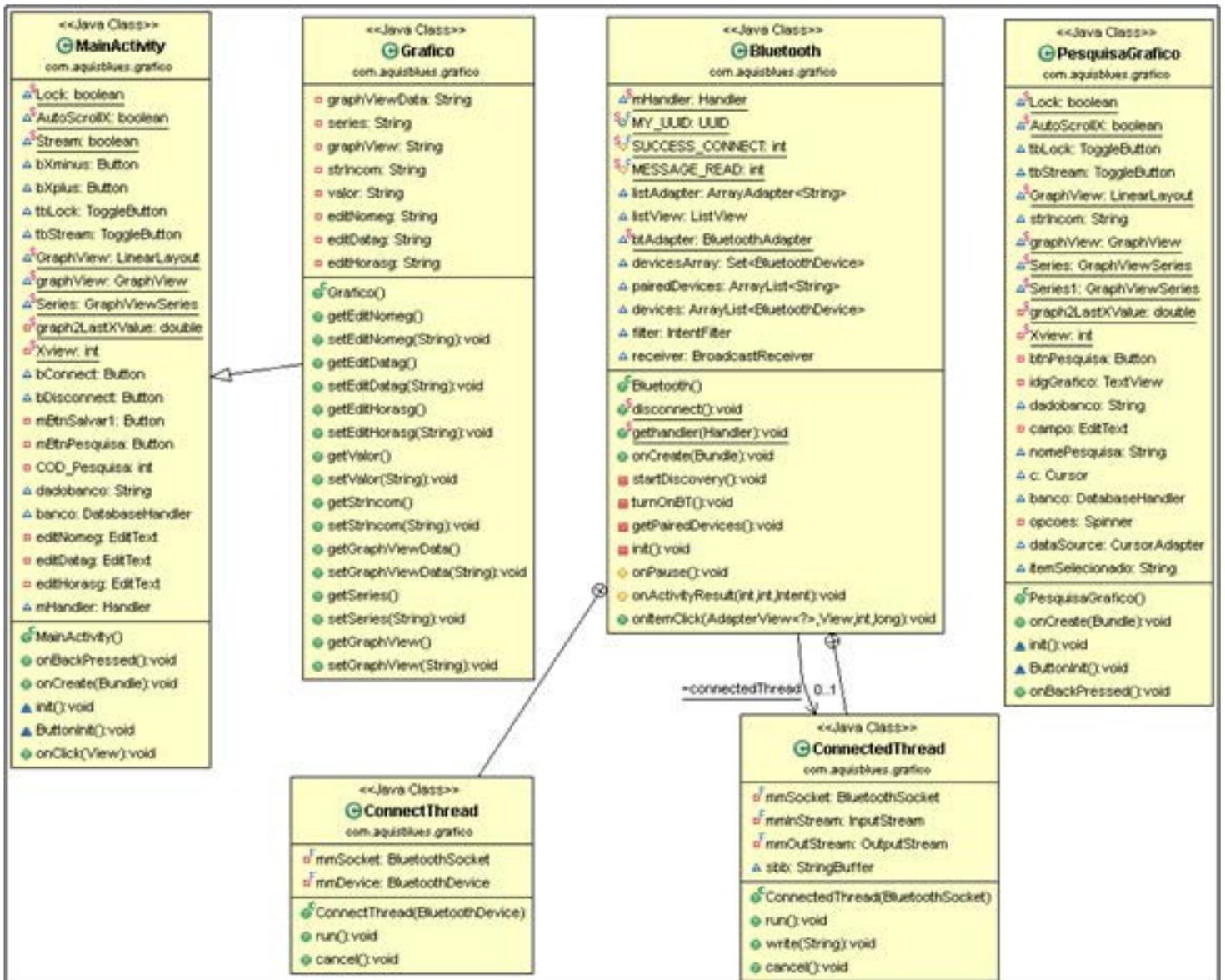
### 5.6.2 Diagrama de Classes

O modelo de classes tem como sua principal característica representar as partes estruturais do sistema, em que cada classe representa uma entidade que, por sua vez, possuem seus respectivos atributos. O projeto do aplicativo em questão foi dividido em três pacotes e cada pacote foi representado pelo diagrama de classe, como pode ser visto a seguir. Na Figura 18, observa-se o diagrama de classe relacionado ao pacote **com.aquisblues**, apresentando o relacionamento das classes e seus atributos em comum.

Figura 18 - Diagrama de classe do pacote **com.aquisblues**.

Fonte: Elaborado pelo autor.

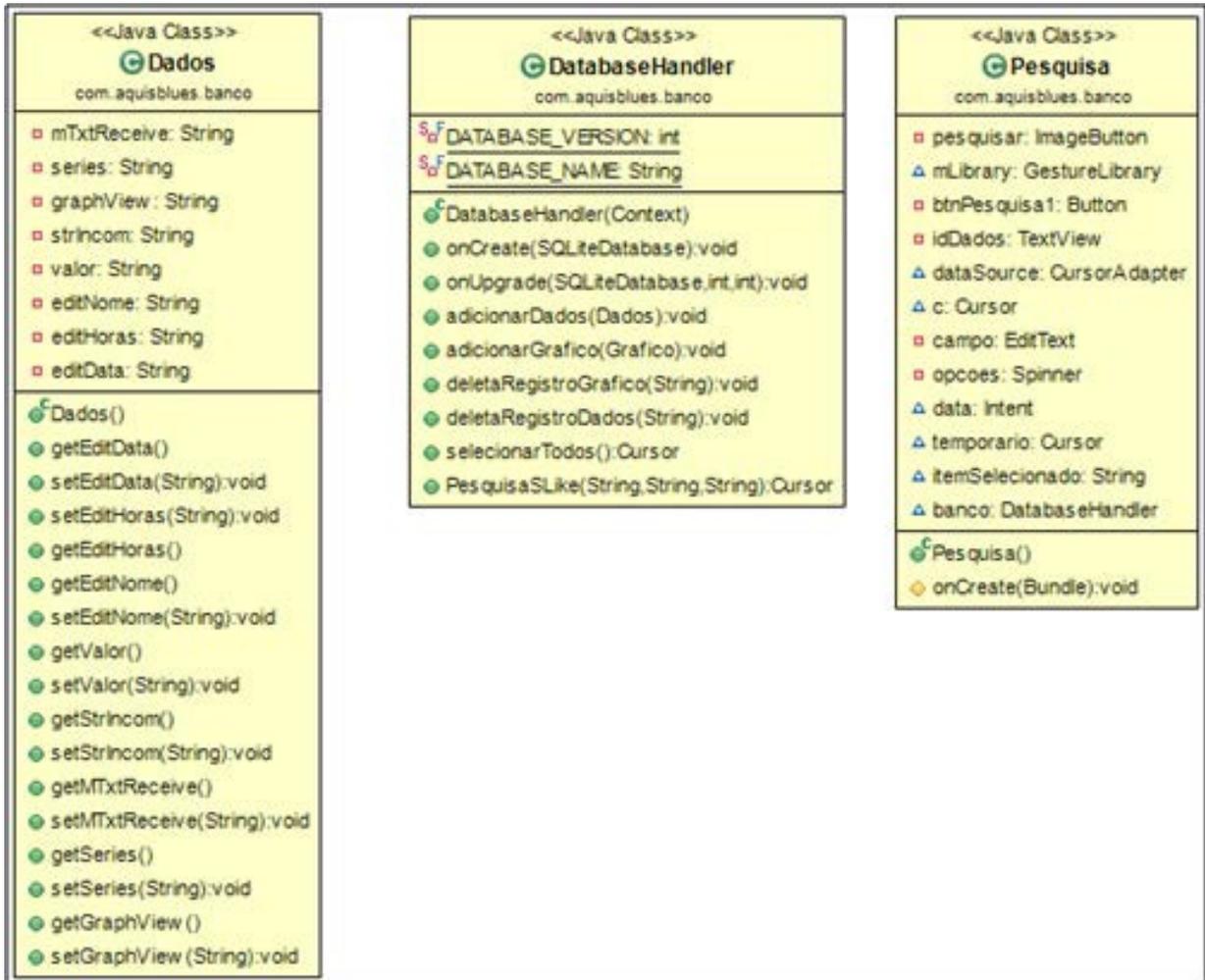
Na Figura 19, apresenta-se o pacote **com.aquisblues.grafico**, demonstrando seu relacionamento entre as classes e atributos em comum, implementado para que fosse possível realizar aquisições de dados gráficos intermediados pela comunicação *bluetooth*.

Figura 19 - Diagrama de classe do pacote **com.aquisblues.grafico**.

Fonte: Elaborado pelo autor.

E, por último, na Figura 20 apresenta-se o relacionamento entre as classes e seus atributos em comum, implementado no pacote **com.aquisblues.banco** do projeto AppAquisBluet, para que fosse possível armazenar os dados em banco com objetivo de futuras análises.

Figura 20 - Diagrama de classe do pacote **com.aquisblues.banco**.



Fonte: Elaborado pelo autor.

### 5.6.3 Diagrama de Fluxo das Telas

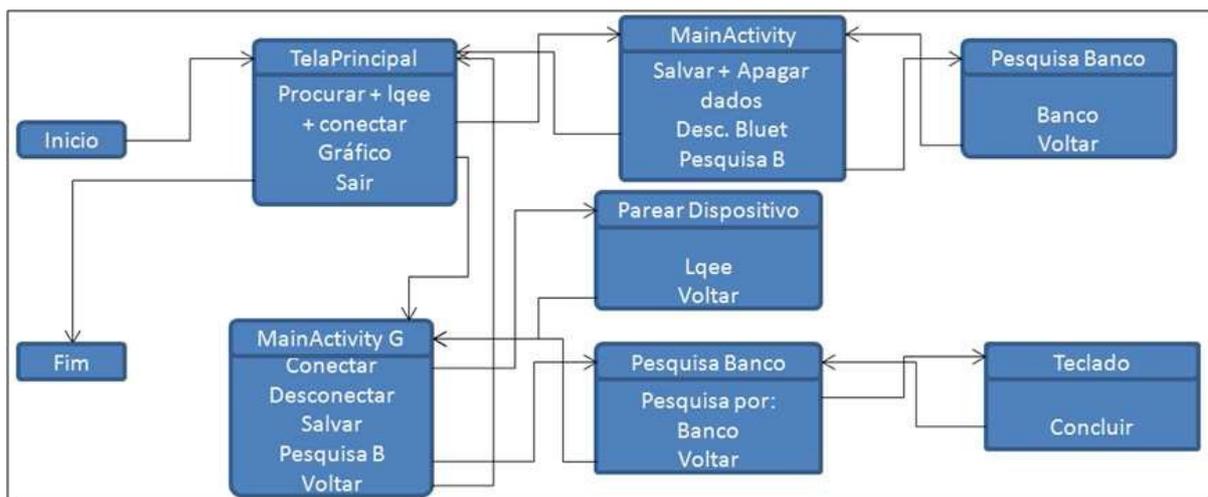
O diagrama de fluxo de telas apresenta o caminho de navegação entre as telas do aplicativo AppAquisBluet, na ordem que aparece para o usuário, da maneira que cada tela seja uma *Activity* da aplicação de aquisição de dados em tempo real. Logo, este diagrama pode ser observado na Figura 21. As telas que apresentam as funcionalidades da aplicação são:

- **Tela principal** - ela se apresenta assim que o aplicativo é iniciado, disponibilizando para o usuário a conexão com o adaptador *bluetooth*, além de apresentar o *layout* da aplicação com suas imagens e botões, que têm a função de chamar a tela *MainActivity* e a tela *MainActivity G* com suas funcionalidades específicas;
- **Tela MainActivity** - tem seus botões com suas funcionalidades relacionada a aquisição de dados e acesso a tela **Pesquisa Banco** com suas funcionalidades voltada a pesquisa dos dados armazenados em banco;

- **Tela MainActivity G** - apresenta o *layout* constituído de botões específicos, cada um com sua funcionalidade para que seja realizada aquisição de dados e plotados graficamente no visor da aplicação e posteriormente, salvos em banco. Além disso, contém relaciona- mento com a tela **Parear Dispositivo**, cujo função é abrir a comunicação *bluetooth* e a tela **Pesquisa Banco**;
- **Tela Pesquisa Banco** - faz uso de alguns campos e botões com objetivo de propiciar para o usuário a pesquisa de dados armazenadas em banco. Relaciona-se com a tela **Teclado**, utilizada para nomear a busca dos dados desejados em banco para uma eventual análise.

Além do diagrama de fluxo das telas, procurou-se apresentar, resumidamente, a função e os relacionamentos de cada tela desenvolvida para o projeto AppAquisBluet.

Figura 21 - Diagrama de fluxo de telas.



Fonte: Elaborado pelo autor.

#### 5.6.4 Estrutura do Banco de Dados

O banco de dados referente a aplicação AppAquisBluet, destinada a aquisição de dados, utilizando o hardware condicionador de sinal via *bluetooth* com ponte de Wheatstone, é constituído por duas tabelas e seus relacionamentos, em que são armazenadas, localmente, as informações adquiridas através da aquisição de dados realizada pelo usuário, visto que toda estrutura e relacionamentos do banco são criados no momento da primeira execução do aplicativo AppAquisBluet. A estrutura do banco pode ser observada na Figura 22, apresentada pela tabela **dados** e tabela **gráfico**.

Figura 22 - Estrutura do banco de dados.



Fonte: Elaborado pelo autor.

Como pode ser observada na Figura 22, tanto a tabela **dados** como a tabela **gráfico**, têm seu **ID** representado pela chave primária, como identificador único relacionado a cada armazenamento de dados recebido pela aquisição via *bluetooth*, e o nome do arquivo, data e hora da aquisição, seja ela gráfica ou não. Logo, ambas as tabelas não possuem qualquer tipo de relacionamento uma com a outra.

## 5.7 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo, procurou-se demonstrar a estrutura do AppAquisBluet desenvolvida na IDE Eclipse, as pastas principais que compõem o projeto e suas contribuições para que a aplicação desenvolvida funcionasse perfeitamente. Além disso, abordou-se a comunicação *bluetooth* e sua implementação no projeto do aplicativo AppAquisBluet para que o mesmo pudesse realizar aquisições de dados intermediada pela comunicação *Bluetooth*, utilizando uma porta serial simulada por RFCOMM.

Outrossim, outros pontos fundamentais foram abordados neste capítulo, a exemplo da implementação do processo de aquisição de dados visualizável em uma interface gráfica em tempo real, administrada pela biblioteca *GraphView* disponibilizada para o Android e o banco de dados *SQLite* para armazenamento, através de um sistema de aquisição de dados, com objetivo de realizar futuras análises, além da modelagem do projeto desenvolvido para o aplicativo AppAquisBluet.

Para o próximo capítulo são apresentados os resultados dos métodos utilizados no desenvolvimento do projeto AppAquisBluet e os resultados dos testes quanto à aquisição de dados em tempo real intermediada pela comunicação *bluetooth*.

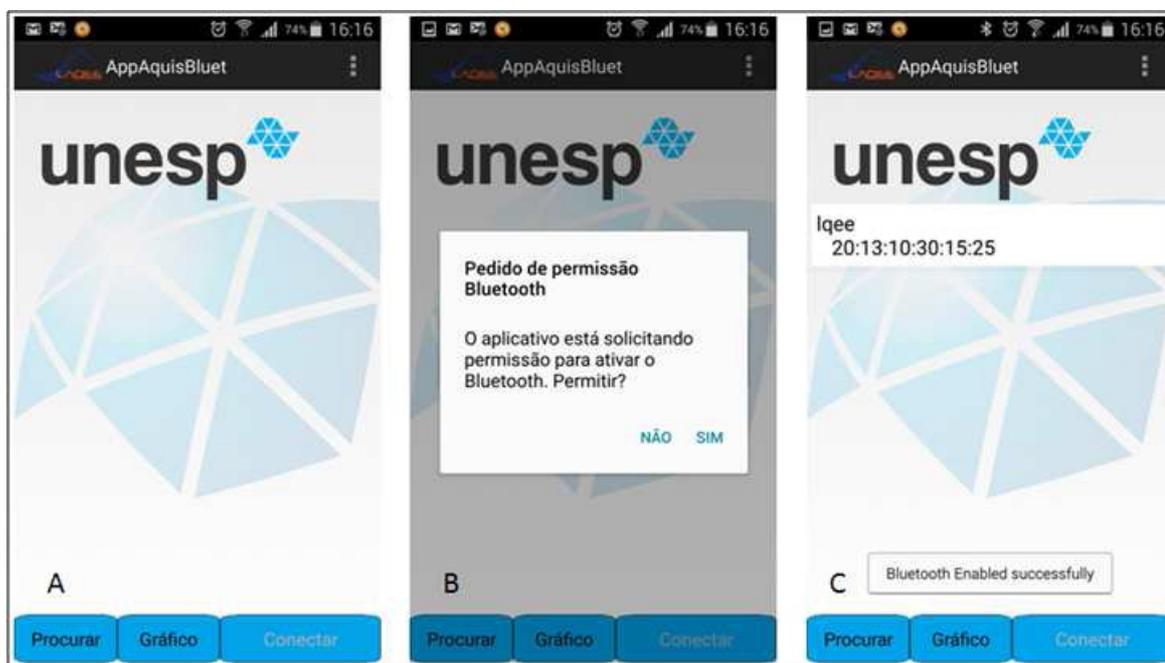
## 6 RESULTADO DOS MÉTODOS UTILIZADOS E AQUISIÇÃO DE DADOS VIA BLUETOOTH

Neste capítulo, são apresentados os resultados dos métodos aplicados no desenvolvimento do AppAquisBluet, tendo como objetivo apresentar a interface do aplicativo e suas funcionalidades. Executaram-se testes de aquisição de dados e plotaram-se os resultados dos dados no gráfico, utilizando a biblioteca *GraphView*. Tais dados foram armazenados no banco *SQLite*, implementado na própria aplicação.

### 6.1 O PROJETO APPAQUISBLUET

O projeto AppAquisBluet ficou constituído de várias pastas, arquivos e bibliotecas que, juntas, formam a estrutura do projeto desenvolvido na IDE Eclipse. Implementaram-se suas *activities*, *intents*, *services*, *broadcasts receivers*, resultando-se na tela principal do aplicativo e suas funcionalidades, como pode ser observado na Figura 23.

Figura 23 – Telas do aplicativo AppAquisBluet: **A**- tela principal; **B**- pedido de permissão *bluetooth*; **C**- adaptador *bluetooth* lqee encontrado.



Fonte: Elaborado pelo autor.

Quando o aplicativo é executado, o usuário se depara com a *activity* inicial (tela principal) representada pela figura com a letra **A**, composta por três botões, em que cada um direciona-se para uma *activity* diferente com funcionalidade específica. Contudo, para que fosse possível

tal funcionalidade, utilizou-se de técnica para inserir os botões, criando um *LinearLayout* sem componentes, contendo apenas os botões com **texto**, utilizando a função *onClickListener* no próprio *LinearLayout*. Assim, pode-se observar, a seguir, a função de cada botão em destaque:

- Acionando em **Procurar** - o usuário ativará a função de procurar por dispositivos que estejam pareados e dentro do raio de alcance do *bluetooth*;
- Acionando em **Gráfico** - o usuário será direcionado para a tela onde será plotado os resultado de aquisição de dados em gráfico;
- Acionando em **Conectar** - o usuário será direcionado à tela de aquisição de dados em tempo real, onde o mesmo poderá inicializar a aplicação propriamente dita.

### 6.1.1 Resultado do pareamento entre o dispositivo e o módulo *bluetooth*

O aplicativo AppAquisBluet só funcionará se o *Bluetooth* do celular estiver ativo. Portanto, a primeira verificação que a *activity* faz, ao ser executada quando se aciona o botão **Procurar**, é saber se o dispositivo Android está com o recurso do *Bluetooth* habilitado. Caso não esteja, uma *activity* central será inicializada, solicitando a ativação do *Bluetooth*. Essa solicitação é automática e nativa de dispositivos Android, como ressaltado no capítulo anterior, sempre que o software requisitar pelo adaptador *Bluetooth* e o mesmo não estiver habilitado. Caso o usuário clique em **Não**, a aplicação será encerrada. Conforme mostrado na Figura 23, pela letra **B**.

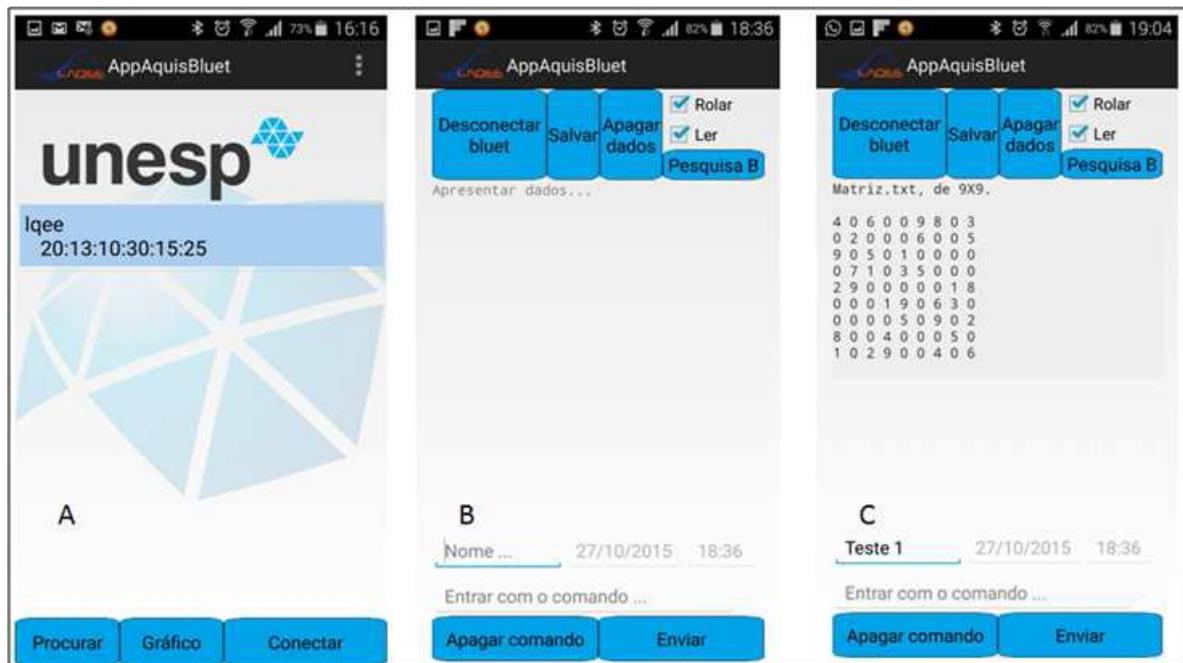
Após permitir a ativação do *bluetooth*, o usuário precisará conectar o celular ao módulo *bluetooth* e iniciar o processo de aquisição de dados. Para isso, é preciso acionar o botão **Procura** e aceitar o pedido de permissão *bluetooth*. A aplicação irá exibir a *DeviceListActivity*, que permite fazer uma procura por dispositivos *bluetooth* disponíveis e selecionar à qual irá se conectar. O módulo *bluetooth* que será conectado está representado como **lqee (20:13:10:30:15:25)** como mostrado na Figura 23, na letra **C**.

### 6.1.2 Tela de aquisição de dados em tempo real

Utilizando-se da API *bluetooth* disponibilizada para o Android, juntamente com seus métodos e classes, criou-se um canal de comunicação entre o dispositivo e o módulo *bluetooth*, realizando, assim, aquisição de dados em tempo real, utilizando a comunicação RFCOMM (*Radio Frequency Communications*), simulando um protocolo de substituição de cabo usado para criar uma porta serial virtual, se tornando um mecanismo para abrir soquetes de comunicação entre dois dispositivos *bluetooth* que estejam pareados para receber e enviar dados.

Na Figura 24, apresenta-se a tela de aquisição de dados, contendo algumas funções específicas, as quais podem ser observadas na tela representada pela letra **B** contendo alguns botões:

Figura 24 - Tela de aquisição de dados: **A**- adaptador *bluetooth* lqee; **B**- funções do aplicativo; **C**- recebendo os dados.



Fonte: Elaborado pelo autor.

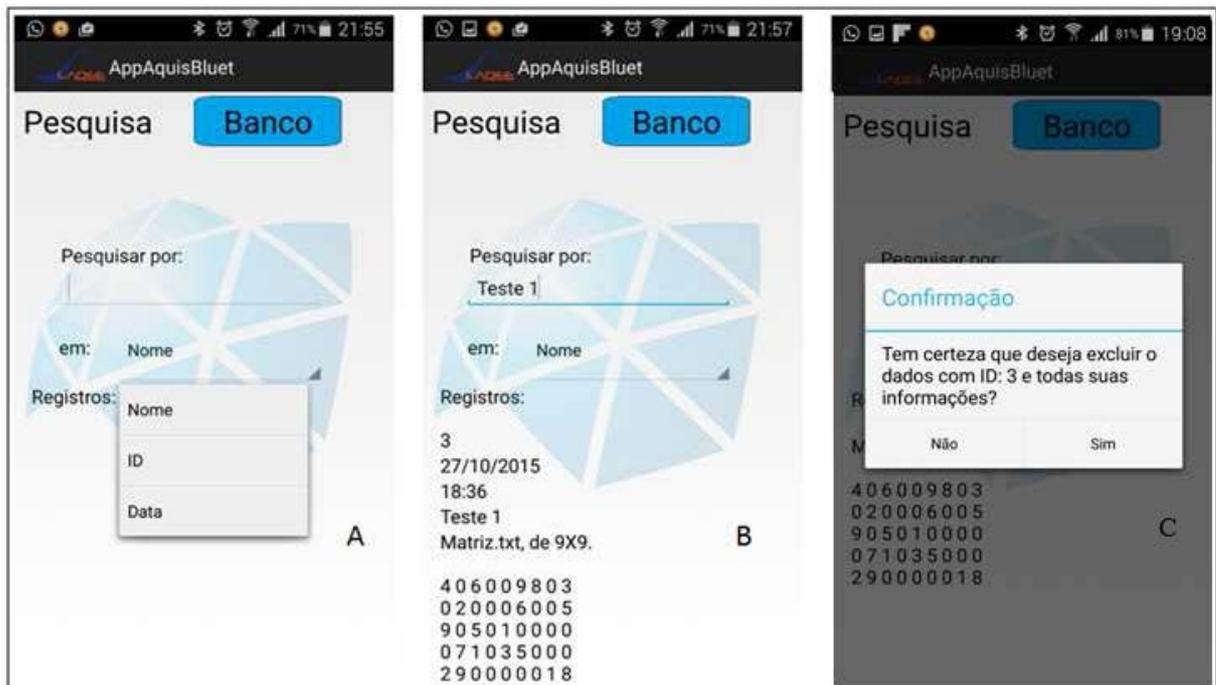
- **Desconectar bluet** - tem a função de encerrar o canal de comunicação entre o dispositivo e o módulo *bluetooth*;
- **Salvar** - tem a função de salvar os dados recebidos no banco *SQLite* da aplicação;
- **Apagar dados** - tem a função de deletar todos os dados recebidos no campo **Apresentar dados**;
- O *checkbox* **Rolar** - tem a função de limitar a visão dos dados na tela da aplicação e o *checkbox* **Ler** tem a função de cancelar a leitura de dados no visor da aplicação;
- **Pesquisa B** - tem a função de direcionar o usuário para uma nova janela na qual poderá consultar dados armazenados em banco;
- Mais abaixo no visor, tem-se o campo, de preenchimento do **Nome** do arquivo recebido, data e horas que serão preenchidos automaticamente pela aplicação;
- Na parte inferior da aplicação, apresenta-se, no visor, o campo onde será digitado o comando, e dois botões, um para envio de comandos para liberação de dado e o outro para apagar comando digitado.

Na tela representada pela letra **C**, da Figura 24, pode-se observar no visor da aplicação uma matriz 9x9, resultado de uma aquisição de dado em tempo real intermediada pela comunicação entre o módulo *bluetooth* e o dispositivo móvel, contexto em que os dados recebidos do módulo *bluetooth* foram armazenados na *String strInput* e apresentado no visor da aplicação.

### 6.1.3 Pesquisando dados no banco

Para realizar uma pesquisa de dados em banco, foram utilizadas técnicas e métodos da linguagem *SQL*, implementando em seu núcleo os elementos básicos e modelos de dados relacionado ao armazenamento de dados. Para o projeto em destaque, utilizou-se o banco *SQLite*, por ser recomendado para aplicações móveis e com funcionalidades que tornam os dados seguros e confiáveis. Como resultado da implementação, apresenta-se na Figura 25, três telas:

Figura 25 - Tela de pesquisa no banco: **A**- pesquisar no banco de dados; **B**- dados encontrados no banco; **C**- deletar dados do banco.



Fonte: Elaborado pelo autor.

- Tela **A** - apresenta o campo **Pesquisar por**, onde o usuário poderá realizar a pesquisa tanto por **Nome**, **ID** (identificador único) ou **Data**, escolhendo a opção desejada. Os resultados estão representados em banco através da chave primária e secundária na coluna da tabela **dados**. Logo abaixo, encontra-se o registro dos dados, do

tipo **arquivo de texto**, armazenado na coluna da tabela **dados** referente à pesquisa. Em seguida, o usuário irá acionar o botão **Banco** e os dados serão apresentados no campo de **Registros**, seguindo a ordem de armazenamento no banco de dados;

- Tela **B** - pode-se observar o resultado da pesquisa realizada por **Nome** e seus respectivos dados, além do **ID**, **Data** e a hora do armazenamento dos dados em banco;
- Tela **C** - apresenta apenas um pedido de confirmação, para excluir dados do banco do respectivo **ID** pesquisado.

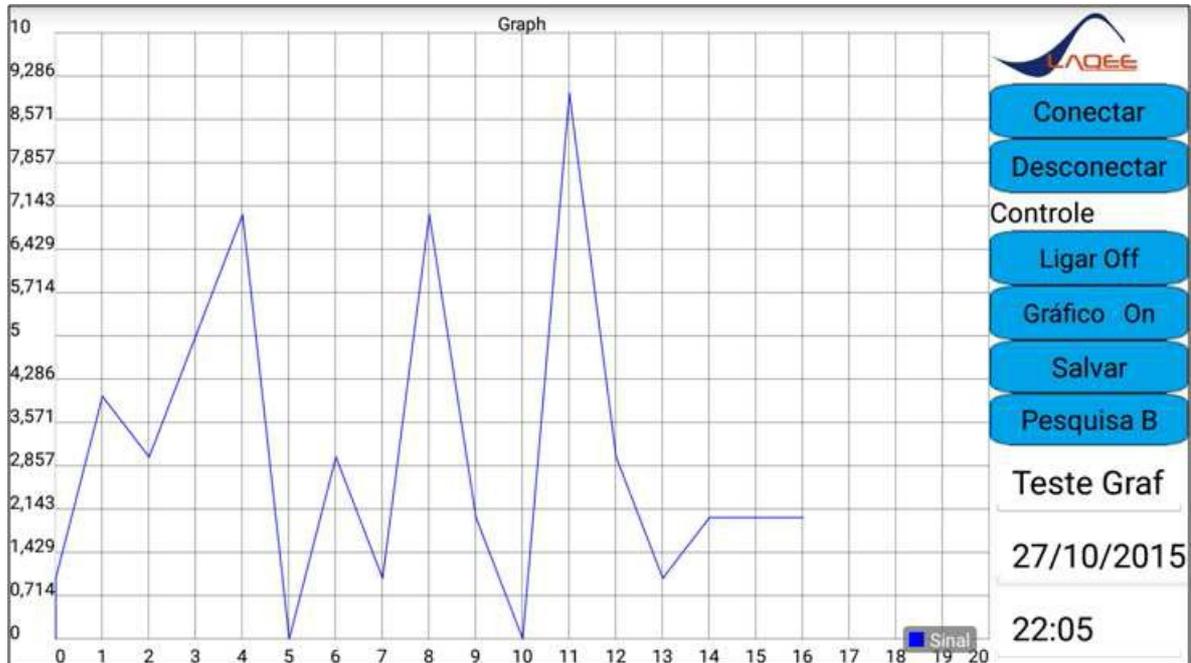
### 6.1.3.1 Pesquisar dados gráficos em banco

Na tela principal da aplicação representada pela Figura 23, da Seção 6.1, tem-se o botão **Gráfico**, com função de direcionar o usuário para uma nova janela, contexto em que será realizada aquisição de dados em tempo real e o resultado será plotado, graficamente, fazendo uso da biblioteca *GraphView*. Como resultado das implementações utilizando métodos, classes e bibliotecas destinadas ao Android, apresenta-se na Figura 26, o eixo **Y** com seu respectivo tamanho de 0 a 10 (valor recebido digital) e o eixo **X** em uma crescente de 0 a 20 pontos. O gráfico pode ser alterado pelo botão **Gráfico On e Off**, além de outras configurações que podem ser observadas na mesma tela:

- O botão de **conectar** e **desconectar** a aplicação ao módulo *bluetooth*;
- O botão de Ligar **Off** e **On**, para liberar a entrada de dados para ser plotado no eixo **Y** e **X**;
- O botão de **Gráfico On** e **Off**, para limitar a entrada de dados;
- O botão **Salvar** irá salvar os dados recebidos em banco para futuras análises;
- O botão **Pesquisa B**, irá direcionar o usuário para uma nova janela, na qual será realizada a pesquisa no banco de dados;
- Abaixo apresenta-se o campo onde será digitado o nome do arquivo para ser armazenado no banco, em que data e hora são preenchidas pela aplicação de forma automática.

Na Figura 26, pode-se constatar o resultado da aquisição de dado sendo plotada no eixo **Y** e **X** com os respectivos valores. Então,  $V = (1; 4; 3; 5; 7; 0; 3; 1; 7; 2; 0; 9; 3; 1; 2; 2; 2)$ , sendo **V** os valores recebidos, aleatoriamente, transmitido pelo módulo *bluetooth*, armazenados em uma *String strIncom* no código da aplicação e plotado um caractere por vez no eixo **Y** com distância contínua de um **ponto por vez** no eixo **X**. Fez-se uso do ponto e vírgula para separar os dados recebidos digitalmente.

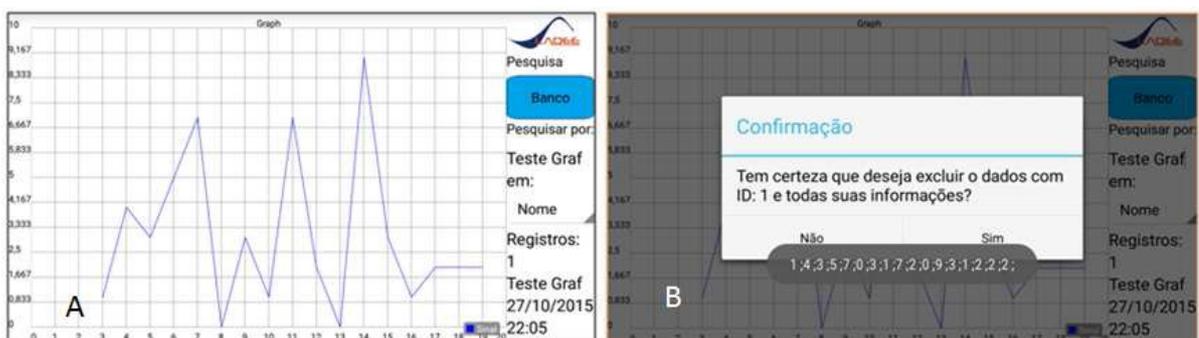
Figura 26 - Aquisição de dados gráficos.



Fonte: Elaborado pelo autor.

Realizando pesquisa no banco, pode-se observar a aquisição de dados demonstrada na Figura 26, apresentada na Figura 27.

Figura 27 - Busca no banco de dados: **A**- resultado da pesquisa no banco; **B**- deletar dados referente a pesquisa.



Fonte: Elaborado pelo autor.

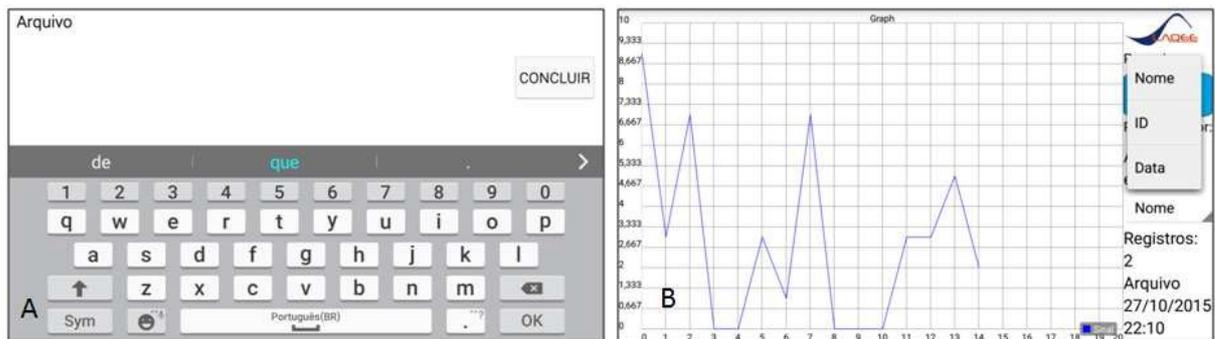
A tela **A** da Figura 27, referencia a tela **Pesquisa Banco**, na qual plotou-se o resultado da pesquisa realizada tanto por **Nome**, **ID** ou **Data** representado no banco através da chave primária e secundária armazenada em suas respectivas colunas da tabela **gráfico** com dados do tipo **texto**.

Preenchendo o campo de **Pesquisa por** e acionando-se o botão **Banco**, o resultado da aquisição armazenada em banco através de **IDs** sequenciais será plotado no visor da tela **A** em uma *ListView* denominada de **Registros**, contendo todos os dados referente a opção de busca. Em seguida, o usuário terá que escolher o dado e acionar sobre ele para, assim, o mesmo ser plotado no eixo **Y** e **X** no visor da aplicação.

Os dados apresentados, na *ListView* **Registros** buscada no banco, podem ser deletados acionando um botão e segurando no dado desejado. Em seguida, irá aparecer um pedido de confirmação para excluir o dado desejado com todas suas informações. Tal confirmação pode ser constatada na letra **B** da Figura 27.

Para realizar uma nova pesquisa de dados gráficos no banco, o usuário terá que digitar a opção desejada entre **Nome**, **ID** ou **Data**, no campo de **Pesquisa por**, utilizando o teclado de digitação do próprio *smartphone* como mostra a imagem na letra **A** da Figura 28.

Figura 28 - Resultado da nova pesquisa em banco: **A**- teclado para digitar a opção de busca; **B**- resultado da pesquisa no banco.



Fonte: Elaborado pelo autor.

Após digitada a opção desejada, o usuário terá que clicar em **Concluir** para armazenar a opção. Em seguida, será direcionado à tela **Pesquisa Banco** para acionar o botão **Banco** e, conseqüentemente, o resultado da opção escolhida será apresentada na *ListView* contendo todos os dados referente à opção de busca. Assim, o usuário acionando o dado listado na *ListView* **Registros**, o mesmo será plotado no eixo **Y** e **X** no visor da aplicação ou acionando e segurando para excluir o dado com todas suas informações contida no banco.

Tal resultado pode ser observado na imagem da letra **B** da Figura 28, com os respectivos valores  $V = (9; 3; 7; 0; 0; 3; 1; 7; 0; 0; 0; 3; 3; 5; 2)$ , em que **V** corresponde aos valores plotados no gráfico e armazenados em banco através da *String dadobanco* e, posteriormente plotado um caractere por vez no eixo **Y** com distância continua de um **ponto por vez** no eixo **X** utilizando (**ponto e vírgula**) como separador de dados.

## 6.2 TECNOLOGIAS UTILIZADAS NA AQUISIÇÃO DE DADOS

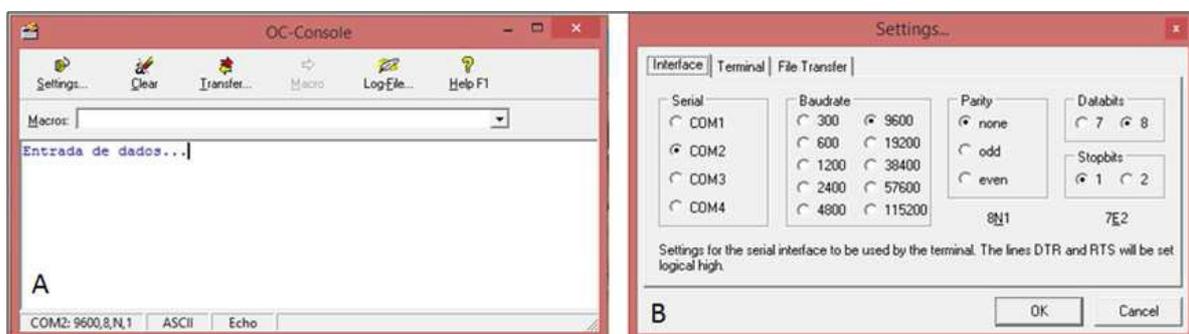
Para realizar os teste de aquisição de dados em tempo real com o aplicativo AppAquisBluet, fez-se uso de algumas tecnologias que intermediaram a aquisição desses dados. Dentre as tecnologias utilizadas, destacam-se o software OC-Console, os hardware *Ultrabook inspiron 14Z*, Samsung Galaxy S5 e o módulo *bluetooth HC-05*, contendo as devidas configurações: o *Ultrabook*, com sistema operacional de 64 bits, Windows 8.1 *Single Language*, processador Intel (R) Core (TM) i5-3337U CPU 1,80 GHz, Memória (RAM) de 4 GB e o *smartphone*: Samsung S5, modelo SM-G900M, versão do Android 5.0 e 16 GB de armazenamento.

### 6.2.1 Software OC-Console

Para realizar os testes de aquisição, utilizou-se a versão *Release 2.5* do OC-Console, tendo a Castlesoft como detentora dos direitos autorais do software. Tal versão é definida como um programa de terminal, cuja tarefa é simular uma porta serial para receber e enviar dados, contendo funções básicas de um programa de terminal com frequência suficiente para satisfazer completamente o desenvolvimento de sistemas embarcados, como: taxa de transmissão selecionável, porta COM selecionáveis e função de transferência de pastas, como pode ser observado na Figura 29.

Na Figura 29, apresenta-se, na tela **B**, a configuração da interface serial, na qual será selecionada uma das quatro porta serial **COM**, e escolhidos os parâmetros para taxa de transmissão, *databits*, *stopbits* e paridade.

Figura 29 - Interface do programa de terminal OC-Console: **A**- tela principal do OC-Console; **B**- ajustes básicos do OC-Console.



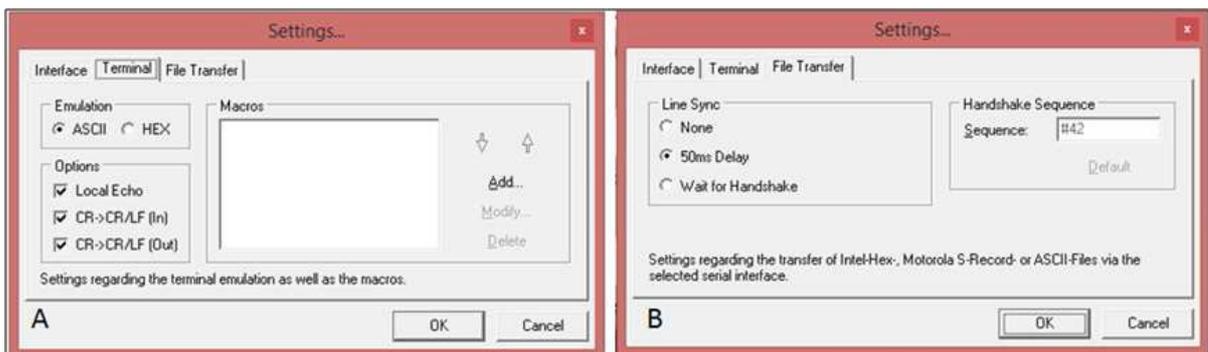
Fonte: Elaborado pelo autor.

Na Figura 30, tela **A**, configurou-se a definição do terminal de diálogo para selecionar a emulação de terminal e para ajustar a formatação do texto, utilizando a emulação ASCII, que mostra caracteres e executa *backspace* ou a emulação HEX, que mostra todos os caracteres

como números hexadecimais e também como texto, se possível utilizando a entrada e saída de dados.

Na letra **B** da Figura 30, é mostrada a configuração dos parâmetros em relação à transferência de arquivos na seção de linha de sincronização, em que será selecionado um dos seguintes modos: *None*, *50 ms Delay* ou *Wait for Handshake*.

Figura 30 - Configuração do terminal e transferência de pasta: **A**- configuração do terminal; **B**- configuração da transferência de dados.

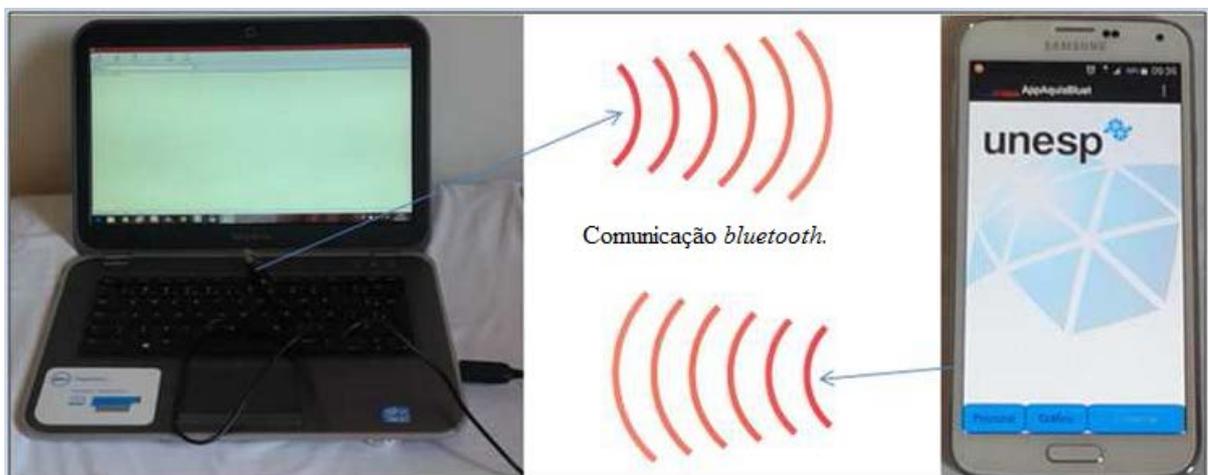


Fonte: Elaborado pelo autor.

## 6.2.2 Estabelecendo a comunicação

Na Figura 31, apresenta-se o computador, o *smartphone* e o módulo *bluetooth* utilizados para simular um sistema de aquisição de dados com transmissão via *bluetooth*, a fim de otimizar os testes em software, sem preocupação com o hardware. Assim, a comunicação entre o programa de terminal OC-Console instalado no *Ultrabook* e o AppAquisBluet instalado no *smartphone* foram concretizadas em sua totalidade.

Figura 31 - Comunicação *bluetooth* utilizando parâmetros da classe 2.



Fonte: Elaborado pelo autor.

Utilizaram-se tais dispositivos por serem de uso restritos do próprio autor. Entretanto, poderiam ser utilizado qualquer sistema, como: Windows, Linux, Mac OS com configurações suficiente para executar o programa de terminal OC-Console e qualquer *smartphones* que execute o sistema operacional Android.

### 6.2.3 Módulo *Bluetooth* HC-05

O módulo *bluetooth* HC-05 (*bluetooth* versão 2.0) transmite e recebe dados através das porta serial TXD (Transmissão de Dados) e RDX (Recepção de Dados), utilizando taxas de 1.200, 2.400, 4.800, 9.600, 19.200 e 38.400 bps. Logo, a taxa definida para realizar os testes relacionado a este projeto foi a 9.600 bps. Assim, pode-se ressaltar que a alimentação do módulo *bluetooth* pode utilizar a tensão entre 3,6 a 6 volts, com antena embutida e atua somente em modo *slave* (escravo).

Para estabelecer a comunicação entre o OC-Console e o *smartphone*, fez-se uma adaptação USB no módulo *bluetooth* para que fosse possível realizar a transmissão de dados através da porta serial. Tal adaptação pode ser confirmada pela Figura 32.

Figura 32 - Módulo *bluetooth* HC-05 utilizado no projeto.



Fonte: Elaborado pelo autor.

## 6.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo, apresentou-se o projeto AppAquisBluet, como produto resultante dos métodos e técnicas utilizadas no desenvolvimento de aplicações móveis disponibilizadas na atualidade. Procurou-se traçar as principais características e funcionalidades da aplicação, além de demonstrar ensaios da aquisições de dados em tempo real, tanto como arquivo de **texto**, ou graficamente no visor da aplicação, haja vista que esses resultados podem ser armazenados em banco e exportados para a memória de armazenamento do *smartphones* para futuras análises.

O próximo capítulo apresenta os resultado dos ensaios, utilizando o aplicativo em questão juntamente com o hardware condicionador de sinais para ponte de Wheatstone intermediada pela comunicação *bluetooth*, de modo que os resultados são apresentados tanto pelo aplicativo como em gráfico trabalhados no software Excel.

## 7 ENSAIOS UTILIZANDO O APLICATIVO APPAQUISBLUET E PONTE DE WHEATSTONE INTERMEDIADA PELA COMUNICAÇÃO *BLUETOOTH*

Após o desenvolvimento do projeto AppAquisBluet, com o uso de técnicas e métodos pertencentes à tecnologia móvel, com objetivo de realizar testes de aquisição de dados em tempo real, em que os dados foram apresentados no visor da aplicação ou graficamente, utilizando a biblioteca *GraphView* intermediada pelo programa de terminal OC-Console, conforme demonstrado nos tópicos anteriores, o próximo teste realizado demonstra tanto a potencialidade do projeto em questão quanto a tecnologia disponibilizada para o desenvolvimento de aplicações móveis destinadas a pesquisas científicas, abordando determinadas áreas no meio acadêmico.

A ponte de Wheatstone foi utilizada por ser um esquema de montagem de elementos elétricos que permite a medição de uma tensão proporcional ao valor de uma resistência elétrica desconhecida, a qual pode ser associada a uma grandeza física (SAMPAIO et al., 1998).

### 7.1 PONTE DE WHEATSTONE

A ponte de Wheatstone foi inventado por Samuel Hunter Christie, no ano de 1833, mas só posto em prática a partir de 1837 por Sir Charles Wheatstone, um físico e inventor britânico, conhecido especialmente por seu trabalho em eletricidade. Nascido em Gloucester, trabalhou de aprendiz, em 1816, com seu tio, um construtor de instrumentos musicais de Londres. Sir Charles era autodidata no campo da ciência, convertendo-se em professor de filosofia experimental da Universidade de Londres no ano de 1834 e, em 1837, juntamente com o engenheiro William Fothergill Cooke, patenteou o primeiro telégrafo elétrico britânico conhecido como ponte de Wheatstone, em homenagem a Sir Charles (HUBBARD, 2013).

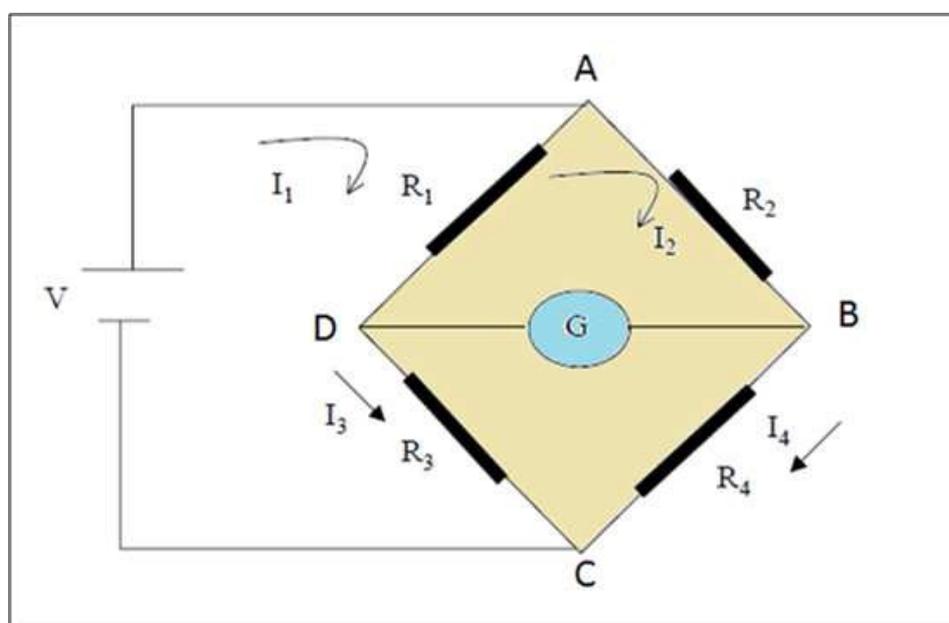
Segundo Sampaio et al. (1998), o circuito elétrico formado pela ponte é utilizado para determinar a variação de uma, ou mais, resistências elétricas. A configuração clássica da ponte é composta por quatro terminais, dois utilizados para alimentação dela e dois para saída de tensão proporcional ao desbalanceamento das resistências elétricas que a compõe. Desta forma, assumindo  $R_x$  como um resistor sensível a variável que se deseja conhecer, e os outros três resistores de valores fixos, em que a variável aplicada ao resistor será proporcional ao valor da resistência desse resistor, a ponte de Wheatstone pode ser aplicada em vários seguimentos, como:

- Medição de Resistência;
- Medição de Temperatura (NTC, PTC);

- Medição de Pressão (*Strain Gage*);
- Medição de Peso (*Strain Gage*);
- Medição de Força/Torque (*Strain Gage*).

Como citado anteriormente a ponte de Wheatstone consiste em dois ramos paralelos de resistências, cada um contendo duas resistências em série. A estas resistências é aplicada uma tensão contínua ou alternada, em casos especiais, de modo a provocar a passagem de correntes nos ramos. Entre os ramos em paralelo, liga-se um detector (galvanômetro), de modo a determinar a condição de balanceamento (corrente nula) (BECK et al., 2009), conforme apresentado na Figura 33.

Figura 33 - Ponte de *Wheatstone*.



Fonte: Elaborado pelo autor.

Quando a ponte está balanceada, isso significa que não passa qualquer corrente no galvanômetro e que o potencial neste é nulo. Nestas condições, isso significa que resistências  $I_1 = I_3$  e  $I_2 = I_4$  e  $I_3R_3 = I_4R_4$ ;  $I_1R_1 = I_2R_2$ , sendo assim, a equação é definida da seguinte forma:

$$\frac{R_1}{R_2} = \frac{R_3}{R_4}, \text{ ou } R_1R_4 = R_2R_3$$

Assim, caso um dos valores da resistência seja desconhecido, é possível determiná-lo por ajuste de valor de uma das resistências do ramo oposto, utilizando a ponte de Wheatstone representada na Figura 33, em que os valores  $R_1 = 15 \text{ k}\Omega$ ,  $R_2 = 10 \text{ k}\Omega$ ,  $R_3 = 30 \text{ k}\Omega$ , que

deverá ser o valor de  $R_4$ , para a ponte tornar equilibrada. De acordo com o que foi apresentado anteriormente, conclui-se, que o resultado é dado pela equação:

$$R_4 = \frac{R_2 R_3}{R_1} = \frac{10 \times 30}{15} = 20 \text{ k}\Omega.$$

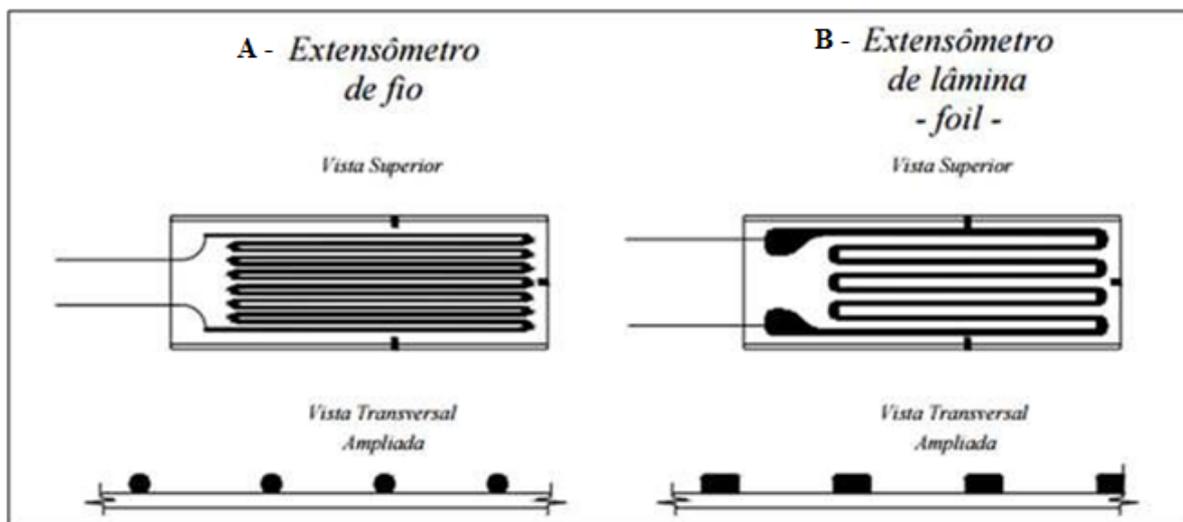
### 7.1.1 Fazendo-se Uso do Extensômetro

Para continuar o projeto em pauta, escolheu-se a ponte de Wheatstone por ser o circuito mais apropriado para converter a mudança da resistência  $\Delta R/R$  de *strain gages*, também denominado extensômetro, para uma tensão de saída  $V_o$ , direcionado à medição de peso.

Por ser um transdutor resistivo, mecanicamente deformável, sua resistência elétrica varia com seu grau de deformação, responsável pela medição da variação da resistência existente no circuito elétrico, o que permite estimar esse grau de deformação, bem como a força aplicada sobre o extensômetro, determinando, assim, a tensão de saída (GUADAGNINI; ROCHA; ELISABETH, 2011).

A definição de extensômetro é dada como **sensor de deformação** pelo fato de *strain* significa deformação, mais precisamente um corpo sob tração mecânica ou compressão; e *gauge* **medidor** (ou sensor), podendo ser encontrado no mercado em variados tamanhos, desde 500 mm de comprimento, utilizados para monitorar dilatação de pontes, edifícios e grandes construções, até unidades menores que 5 mm, utilizadas na odontologia para registrar dilatações em dentes humanos. Existem dois tipos básicos de extensômetros, o *wire gage*: **extensômetro de fio**; e o *foil gage*: **extensômetro de lâmina**, como pode ser observado na Figura 34.

Figura 34 - Dois tipos básicos de extensômetros: **A**- de fio; **B**- de lâmina.

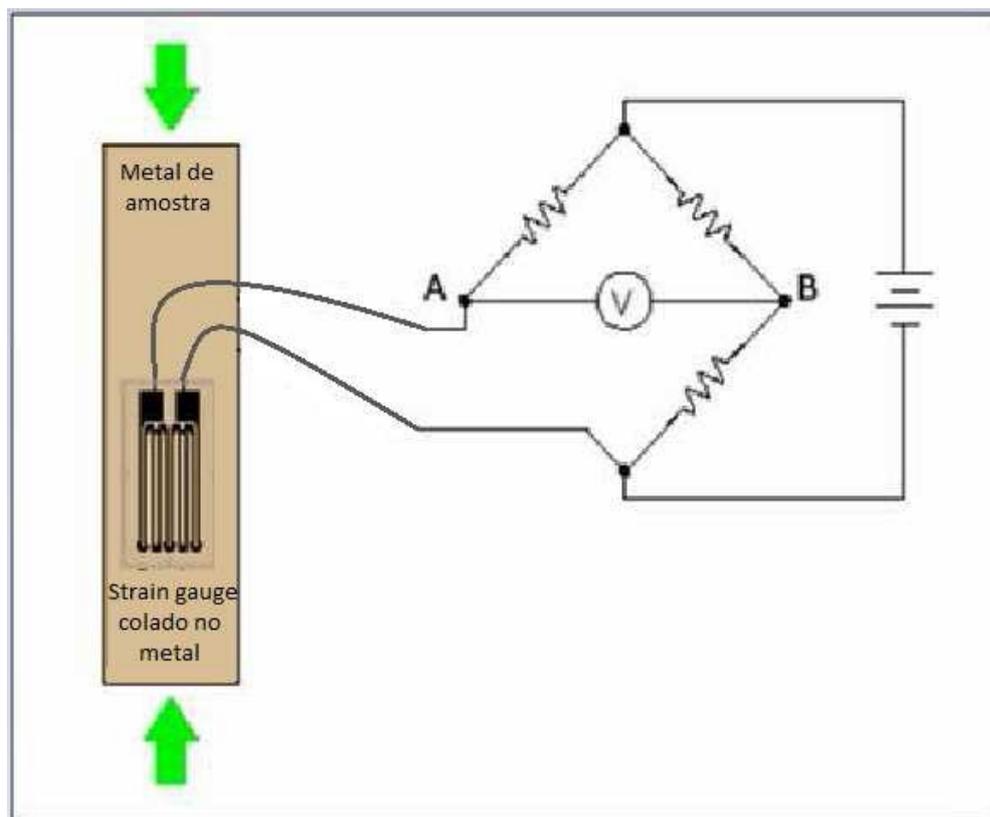


Fonte: (ANDOLFATO; CAMACHO; BRITO, 2004).

O material utilizado na produção do extensômetro é semelhante ao do filme fotográfico (*polimida*), e fixado sobre a superfície a ser analisada, utilizando cola específica. Dessa forma, o eixo extensométrico do sensor dilata-se juntamente com ela, e o material resistivo altera seu valor ôhmico linearmente. Através da variação de resistência elétrica exercida, o sistema eletrônico é capaz de quantificar a dilatação ocorrida (ANDOLFATO; CAMACHO; BRITO, 2004).

Como exemplo de aplicação da ponte de Wheatstone, pode-se citar a utilização em células de carga instrumentadas, fazendo-se uso de extensômetros. Nesta montagem, em uma aplicação prática, o usual é quatro extensômetros que, submetidos a uma carga axial, se deformam mudando o comprimento, a área da seção transversal do elemento, e também a resistência. Na Figura 35 apresenta-se o circuito elétrico do extensômetro com a ponte de Wheatstone, demonstrando que, à medida que a força de compressão aumenta, a ponte fica não balanceada, passando o voltímetro a indicar uma medida não nula, independente da natureza da força utilizada inicialmente.

Figura 35 - Circuito elétrico do extensômetro com a ponte de Wheatstone.



Fonte: Elaborada pelo autor.

## 7.2 CONDICIONADOR DE SINAL COM TRANSMISSÃO DE DADOS VIA *BLUETOOTH* COM PONTE DE WHEATSTONE

O condicionador de sinal não foi desenvolvido especificamente para este trabalho, posto que pode ser utilizado em vários segmentos: medida de resistência, pressão, temperatura, força/torque e medição de peso. Para este estudo, implantou-se o transdutor de peso utilizado como balança em forma de barra, desenvolvido sob medida, construídos na Universidade Estadual Paulista (UNESP Câmpus de Ilha Solteira), no LAQEE (Laboratório de Qualidade de Energia Elétrica) bem como todos os testes e calibração.

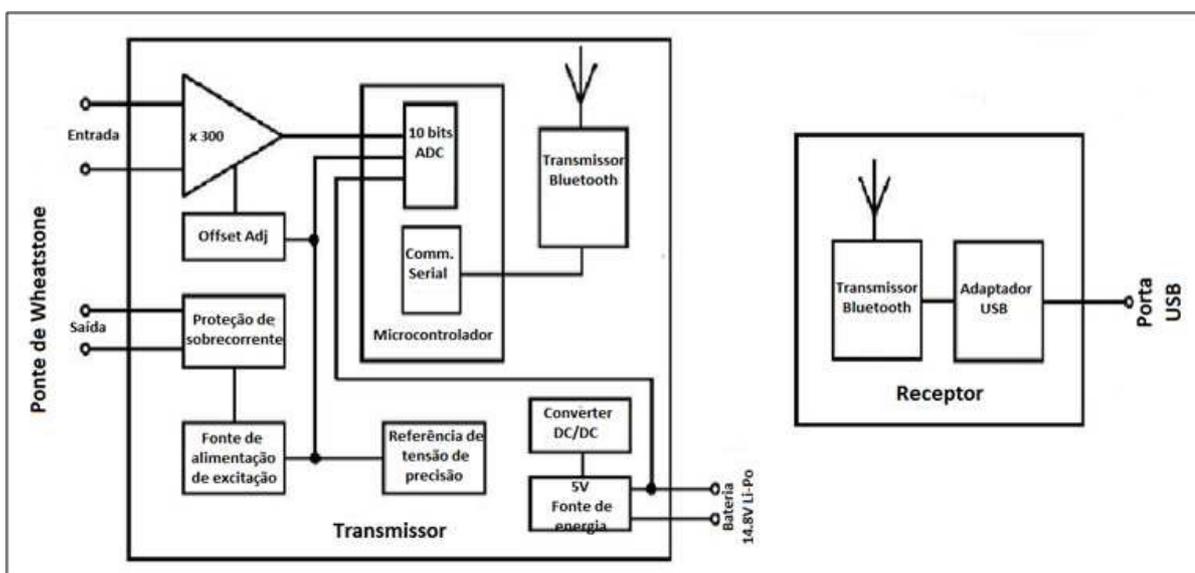
O peso (massa) instantâneo fornecido pela balança instrumentada é medido, digitalizado e, em seguida, transmitido através de um dispositivo sem fios. Logo, o sinal transmitido é captado por um receptor *bluetooth* do *smartphone*, utilizando aplicativo AppAquisBluet, desenvolvido, particularmente, para esse estudo com objetivo de realizar aquisições de dados em tempo real intermediada pela comunicação *bluetooth*.

Nesse sentido, a ligação sem fio é realizada usando a tecnologia *bluetooth*, o que leva a uma ligação de dados segura e rápida. Tal tecnologia possui ampla utilização, visto que

outros dispositivos podem ser utilizados para obter os dados de medição, comunicando-se com o aplicativo AppAquisBluet.

Durante o desenvolvimento do trabalho, um receptor foi utilizado para aumentar o alcance sem fio e permitir que a unidade de aquisição possa ser colocada em local estratégico, priorizando a segurança do usuário, se necessário. O diagrama de blocos do receptor e condicionador de sinal podem ser vistos na Figura 36.

Figura 36 - Diagrama de blocos do receptor e condicionador de sinal.

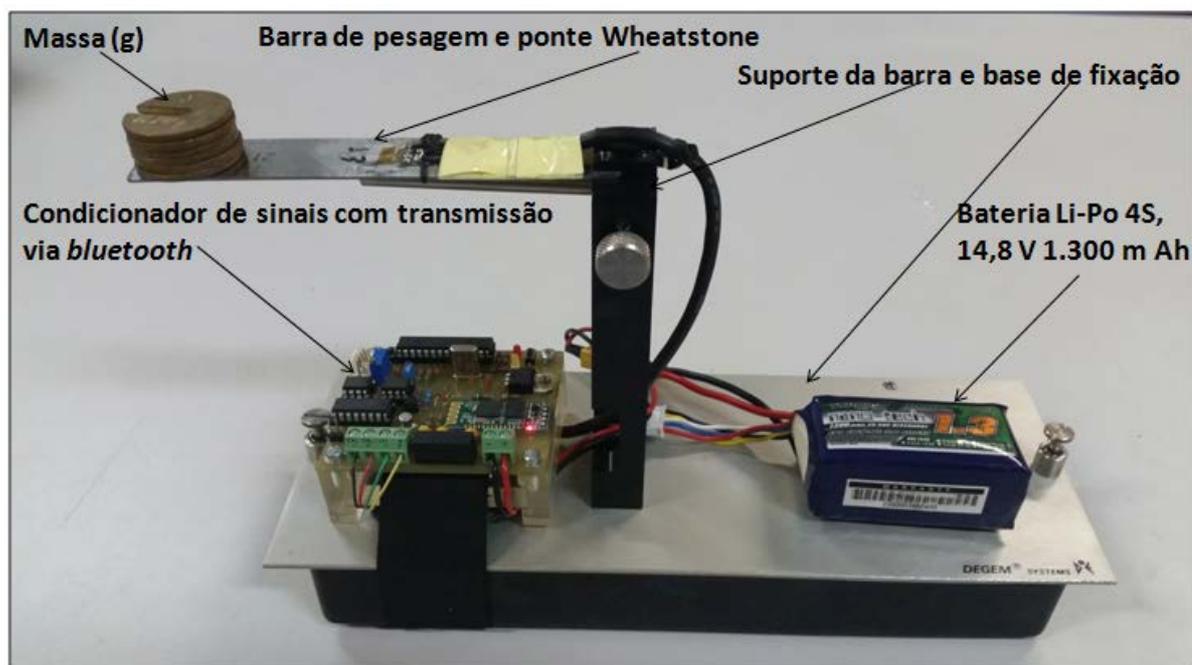


Fonte: Adaptado de (OLIVEIRA, R. N, 2015 ).

### 7.2.1 Condicionador de Sinal com Extensômetro Fixado na Barra de Pesagem

Como pode ser visto no diagrama de bloco e na Figura 37, o circuito pode ser dividido em alguns subconjuntos, dos quais os mais importantes:

Figura 37 - Condicionador de sinal via *Bluetooth*.



Fonte: Elaborado pelo autor.

- Barra de pesagem: sendo a ponte de Wheatstone completa, formada por quatro extensômetros. Logo, dois inferiores e dois na parte superior da barra de pesagem, conectados ao condicionador de sinal, para alimentação e amplificação;
- Amplificador de entrada: um amplificador com ganho fixo de 300;
- Tensão de excitação da ponte fixa em 10 V;
- Tensão referência de : 5 V fixo, estável quanto à variação de temperatura;
- Microcontrolador: responsável pela conversão A/D e controle do *bluetooth*;
- Transceptor *bluetooth*: ligação sem fios *bluetooth*.

Como o ganho de entrada é fixado em 300, calculado para corresponder às necessidades deste teste particular, a tensão máxima de entrada é 16,67 mV, o que fornece um sinal de 5 V para o *Analog to Digital Converter* (ADC). A tensão de excitação também é fixa e calculada para uma ponte de Wheatstone construída com extensômetros de tensão 350  $\Omega$  ohms, neste caso, 10 V. Esta tensão de alimentação também possui uma proteção contra sobrecorrentes, para evitar danos em caso de curto-circuito de sua saída, em que a corrente máxima é limitada a 60 mA.

A tensão da bateria também é monitorada, de modo que se cair abaixo do valor operacional, um LED vermelho será ligado, indicando que a bateria necessita de substituição.

### 7.2.2 Formato dos Dados do Condicionador de Sinal

Os dados são transmitidos em formato serial através do *bluetooth*, lida pelo aplicativo AppAquisBluet, utilizado para receber a aquisição de dados intermediada pela comunicação *bluetooth*, depois que o pareamento do condicionador de sinal com o *smartphone* for liberado. Os dados são **trafegados** como palavras de 10 bits, variando de 1 a 1024. Para uma medição de peso, o *offset* é ajustada em 512.

Quanto às especificações gerais do condicionador de sinal, pode-se observar:

- Taxa de transmissão: 9600 bps;
- Tensão de entrada analógica x digital: 0 mV - 0; 16,67 mV – 1.024;
- Tensão de excitação: 10 V;
- Corrente de excitação máxima: 60 mA (protegida);
- Resposta de frequência: 0-60 Hz;
- Consumo de corrente total: 145 mA (extensômetro de 350  $\Omega$ );
- Alcance sem fio: 10 m;
- Tipo de bateria: *Li-Po* 14,8 V 1.300 mAh.

### 7.2.3 Calibração do Transdutor de Pesagem

A calibração define-se como um conjunto de operações que estabelece, em condições específicas, a correspondência entre o estímulo e a resposta de um instrumento de medir. Sendo um sistema de medição ou transdutor de medição, permite determinar um ou mais parâmetros da curva característica que relaciona o estímulo à resposta ou os valores de grandezas correspondentes às divisões de escalas indefinidas de um instrumento de medir.

Tem por objetivo descobrir, experimentalmente, o **erro sistemático** (uma espécie de desvio constante) e a máxima **incerteza de medição** associada a um determinado instrumento, contexto no qual a **incerteza de medição**, de maneira bastante simplificada, é uma medida estatística da qualidade dos resultados apresentados por um determinado instrumento. Assim, um instrumento com baixa **incerteza de medição** terá a tendência de apresentar resultados mais próximos ao valor verdadeiro da medição.

Sendo assim, a calibração foi realizada com quatro parâmetros de medida, em que a entrada foi gradativamente aumentada e diminuída para que se pudesse chegar no **dado corrigido**. Na tabela 7, podem-se observar a variação dos **massas adicionadas**, **massa total**, **dado recebido** e o resultado final do **dado corrigido**.

Tabela 7 - Calibragem do transdutor de pesagem.

Massas Adicionadas	Massa Total	Dado Recebido	Dado Corrigido
0,000	0,000	512	0
12,100	12,100	522	10
12,080	24,180	533	21
12,285	36,465	543	31
12,285	48,750	553	41
12,060	60,810	563	51
12,350	73,160	573	61
12,442	85,602	583	71
12,355	97,957	594	82
12,260	110,217	604	92

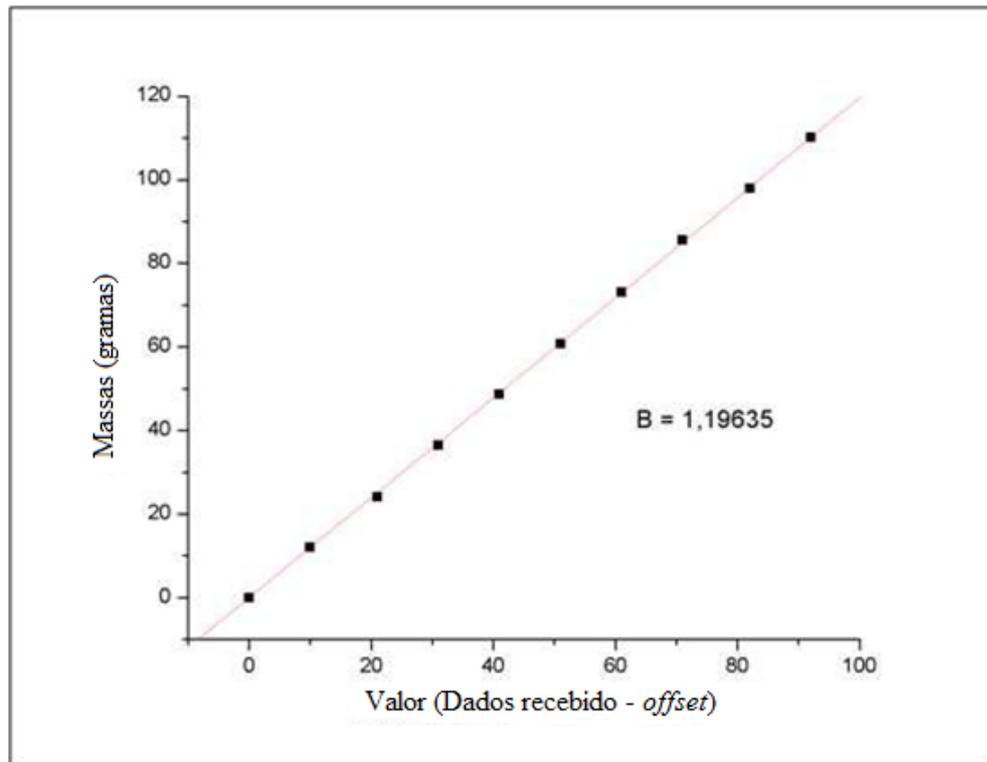
Fonte: Elaborado pelo autor.

Para se chegar ao valor de calibragem aceitável, fez-se uso da equação genérica: **Resultado = (Dado Recebido - *offset*) x Fator de Escala**, demonstrando-se os erros relativos obtidos pela diferença entre cada medida e seu valor final dado pela equação **Massa (g) = (Dado Recebido - 512) x 1,196**, de modo que o *offset* para esse condicionador de sinal foi determinado em **512** e o fator de escala ficou numa margem de erro de **1,196**, valor que pode variar de sensor para sensor, dependendo do processo de calibragem.

A linearidade de um sensor é um tipo de parâmetro que expressa o quanto a sua curva característica se desvia de uma reta de calibração, característica típica de equipamentos ou sensores cuja relação entre entrada e saída pode ser considerada linear. Neste caso, o fabricante especifica uma reta de calibração para o equipamento, definida de três formas diferentes, como: a reta que passa pelos pontos extremos da curva de calibração média, a reta que minimiza o erro com a curva de calibração média ou a reta que passa pela origem e minimiza o erro com a curva de calibração média.

Na Figura 38 apresenta-se o fator de escala dado pela reta linear, de calibração escolhida (curva média geral), para determinar o fator de escala aceitável.

Figura 38 - Fator de escala.

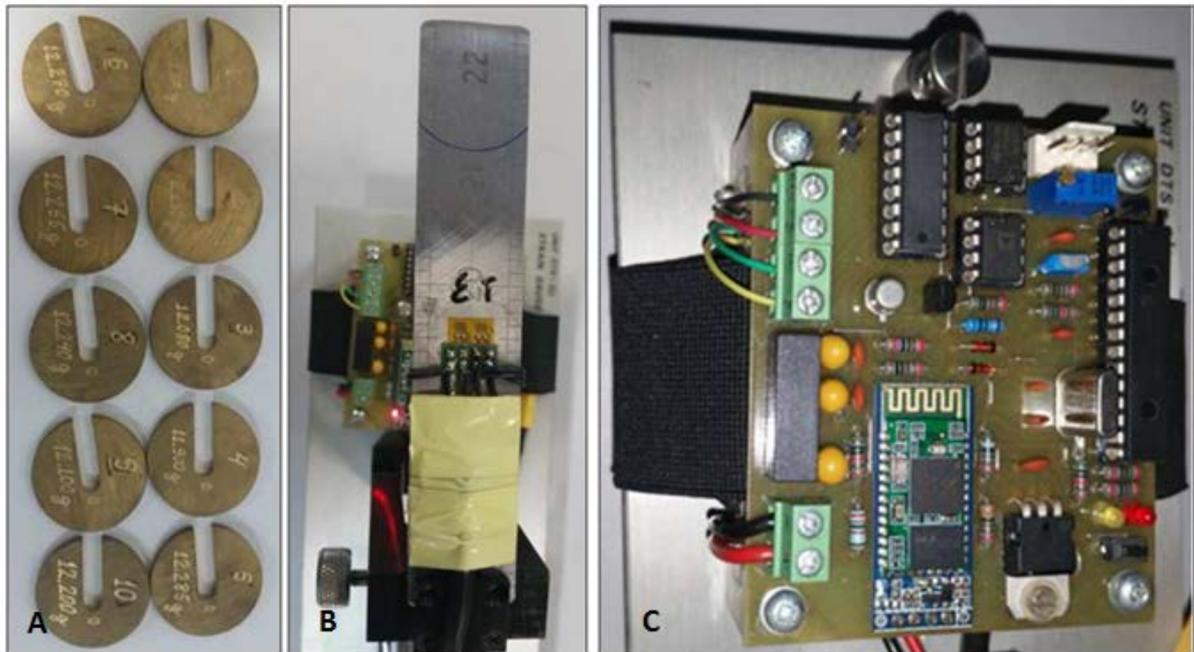


Fonte: Elaborado pelo autor.

Existem vários ajustes que podem ser feitos em sistemas de medição e dentre eles pode-se citar o ajuste de zero (*offset*), ajuste de ganho ou sensibilidade. Para o projeto em questão, trabalhou-se com o ajuste *offset*, como pode ser observado na Figura 38, feito para tornar a saída do sistema de medição igual a zero, quando a entrada for nula. Com isso, não pode se confundir o processo de ajustes com calibração, pois a calibração é um pré-requisito para o ajuste.

Pode-se observar, na Figura 39, alguns componentes que fazem parte do transdutor de pesagem, desenvolvido para realizar aquisições de dados em tempo real via *bluetooth*.

Figura 39 - Componentes do condicionador de sinal: **A**- pesos utilizados; **B**- barra de pesagem; **C**- condicionador de sinal.



Fonte: Elaborada pelo autor.

A Figura 39 é composta por três blocos de imagens, sendo:

- O bloco **A** representa os pesos utilizados como entrada de dados em variadas medidas;
- O bloco **B** demonstra a barra de pesagem com a marcação adequada para inserir os pesos e o extensômetro fixado, projetado para converter deformações lineares em sinais elétricos de entrada acoplado a ponte de wheatstone;
- O bloco **C** representa o condicionador de sinal, composto por seus componentes, tendo como objetivo converter e realizar a transferência dos dados recebidos para o aplicativo AppAquisBluet fazendo uso da comunicação *bluetooth*.

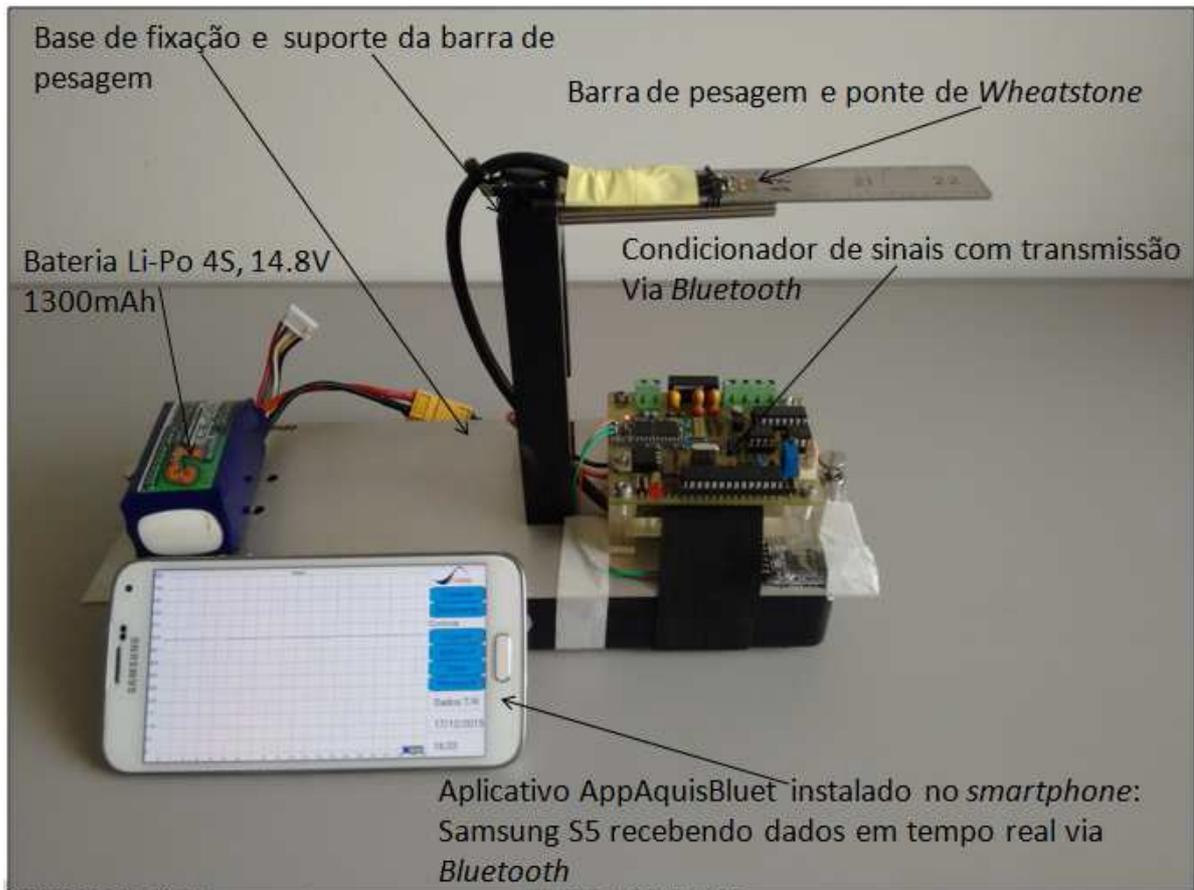
### 7.3 RESULTADOS DA AQUISIÇÃO DE DADOS REALIZADA COM O CONDICIONADOR DE SINAL VIA *BLUETOOTH* UTILIZANDO A BARRA DE PESAGEM COM PONTE DE WEATSTONE

Nesta seção têm-se os resultados finais proposto para este trabalho, de modo que é apresentada, como produto resultante, a medição da variação da tensão em função da deformação sofrida pelo extensômetro que compõe a ponte de Wheatstone fixada na barra de pesagem, em que são colocados os pesos de variadas massas e transmitidos os respectivos valores ao condicionador de sinal para ser convertido para valor digital e enviado via *bluetooth* para o aplicativo AppAquisBluet, apresentando o resultado da aquisição no visor da aplicação

tanto arquivo de texto ou graficamente.

Na Figura 40, pode-se observar a aquisição de dados sendo transmitida em tempo real, intermediada pela comunicação *bluetooth*, apresentado no visor do *smartphone* o *offset* para este ensaio.

Figura 40 - Aplicação recebendo dados.



Fonte: Elaborado pelo autor.

### 7.3.1 Procedimento Para Realizar os Ensaios

Para realizar os ensaios envolvendo o condicionador de sinal via *bluetooth* e o aplicativo AppAquisBluet, foi levantada a reta de calibração do condicionador de sinal para determinar o *offset* para esta etapa do trabalho e, posteriormente, realizou-se o ajuste do fator de escala para chegar-se no valor real do peso (gramas) adicionado na barra de pesagem.

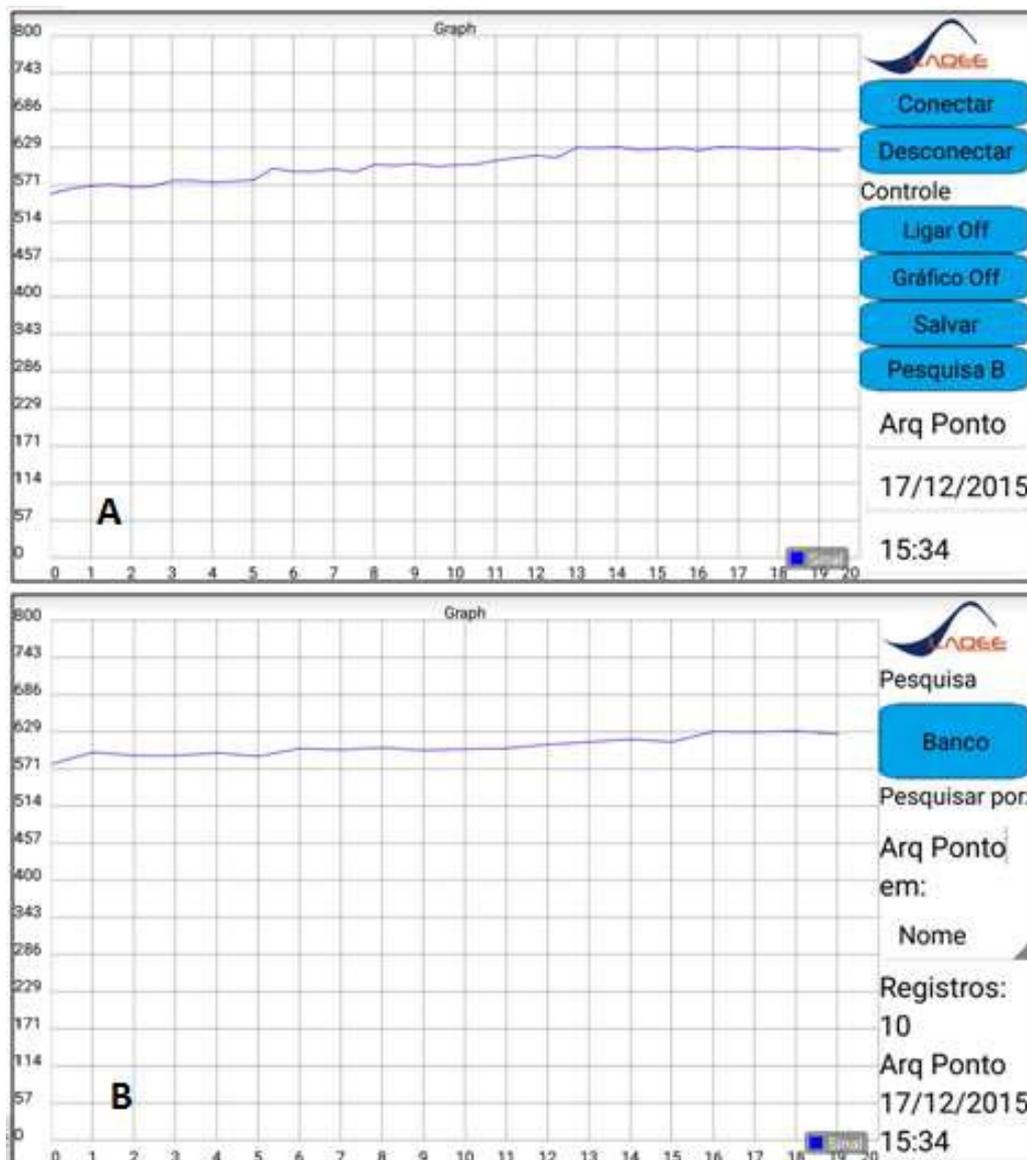
O resultado da aquisição dos dados é apresentado na tela do aplicativo constando apenas quatro caracteres, tendo início pelo *offset* com tempo estipulado de um segundo entre os dados recebidos. Após o recebimento dos dados, tanto em arquivo de texto ou pontos gráficos, acionando o botão **Salvar** do aplicativo, a aquisição realizada será salva no banco SQLite, além de gerar um arquivo de texto na memória de armazenamento do *smartphone*,

possibilitando futuras análises dos dados relacionados à aquisição.

Os pesos (gramas) apresentados na Figura 39, foram colocados na barra de pesagem de forma aleatória, de modo que o tempo de inserção de um peso para o outro foi determinado em três segundos para que fosse possível visualizar a disparidade dos valores apresentados tanto no gráfico como em texto. Logo, na Figura 41 apresenta-se o resultado do ensaio com pesos aleatório representados graficamente.

- Na tela **A** da Figura 41, podem-se observar os valores da aquisição plotados no visor da aplicação;
- Na tela **B** demonstra-se o resultado da pesquisa em banco relacionado à aquisição, plotando os valores armazenados em banco na tela do aplicativo;

Figura 41 - Ensaio com pesos aleatórios: **A**- aquisição de dados; **B**- pesquisa no banco.



- O resultado da tela **C** da Figura 42 apresenta os valores relacionados à aquisição de dados armazenados em banco e exportados como arquivo de texto para a memória de armazenamento do *smartphone*. Sendo assim, realizou-se tratamento dos dados no software Excel e apresentou-se graficamente o resultado dos dados;
- E, por fim, a tela **D** representa os dados do arquivo de texto gerado e exportado para o *smartphone*, contendo as devidas informações da aquisição realizada em tempo real, utilizando o condicionador de sinal via *bluetooth*.

Figura 42 - Ensaio com pesos aleatórios tratado no software Excel: **C**- aquisição de dados; **D**- arquivo de texto gerado. Eixo Y= *offset* mais massa (gramas).



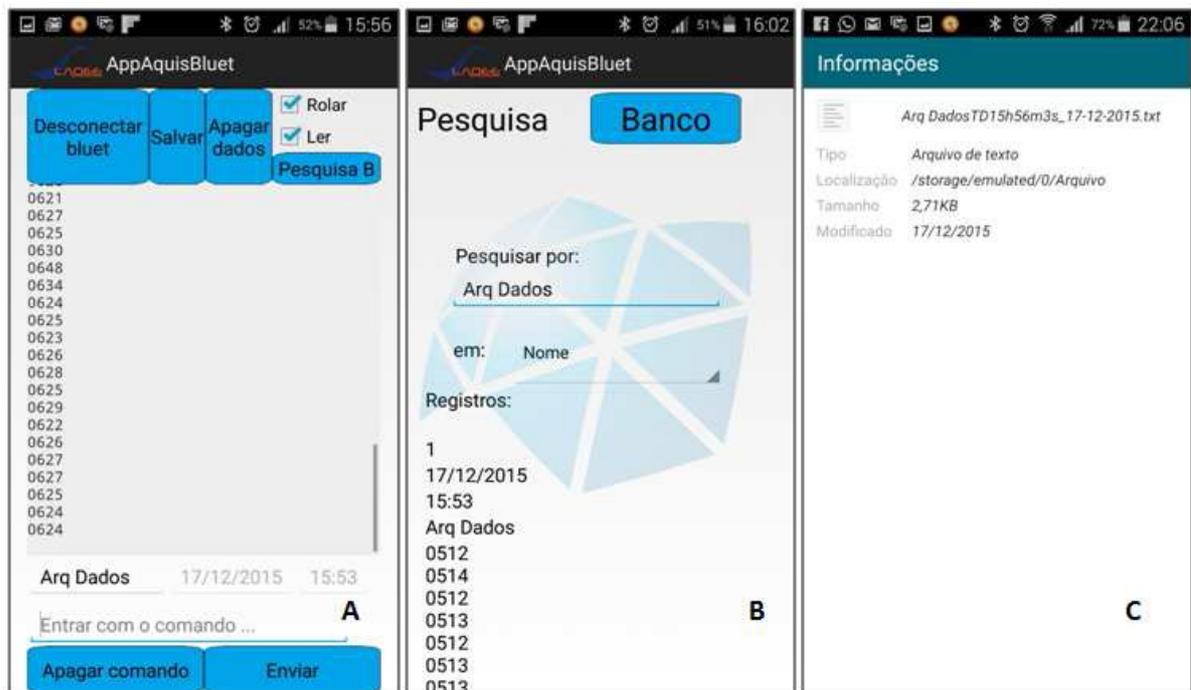
Fonte: Elaborado pelo autor.

### 7.3.2 Aquisição de Dados Exibida em Texto

O aplicativo AppAquisBluet disponibiliza para o usuário os recursos para realizar aquisições de dados em tempo real, plotando tal resultado graficamente em uma tela específica, utilizando as funções da biblioteca *GraphView*, além de possibilitar o arquivamento dos resultados em formato texto, a partir das funções e métodos da programação Android específicos para a comunicação *Bluetooth*. Na Figura 43, pode-se observar o resultado de uma aquisição do tipo **texto**, apresentada com apenas quatro caracteres com intervalo de um segundo de tempo de um dado recebido para outro.

- Na tela **A** da Figura 43 apresenta-se a aquisição realizada em tempo real, com dados numa coluna contínua;
- Na tela **B** demonstra-se o resultado da aquisição salva em banco para futuras análises, contendo nome do arquivo, data e hora da aquisição;
- E, por fim, na tela **C** apresenta-se as informações relacionadas ao arquivo de texto criado e armazenado na memória do *smartphone*, possibilitando ao usuário exportar o resultado da aquisição para um email ou alguma base de dados específica.

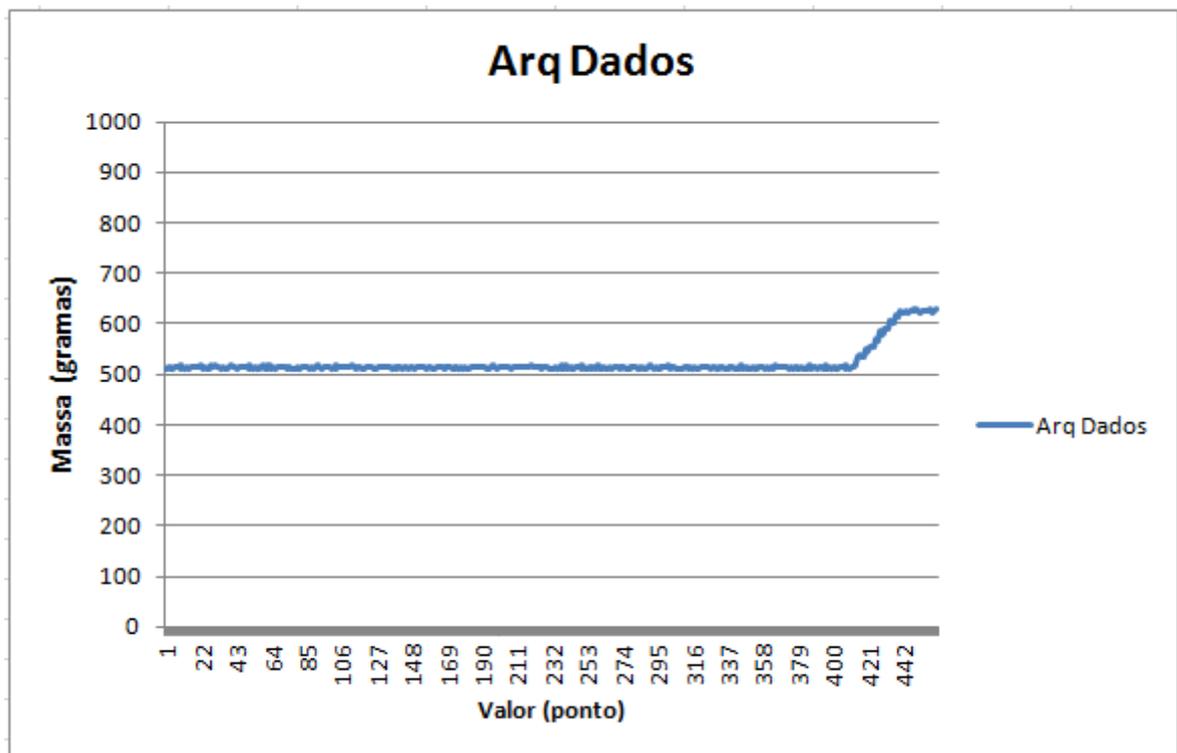
Figura 43 - Ensaio com pesos aleatórios apresentando dados do tipo texto: **A**- aquisição de dados; **B**- pesquisa em banco; **C**- informações do arquivo de texto.



Fonte: Elaborada pelo autor.

Após ter realizado a aquisição dos dados, o usuário, ao acionar o botão **Salvar os Dados**, terá salvos no banco e na memória de armazenamento do smartphone, gerando o arquivo de texto para futuras análises. Na Figura 44 apresenta-se os dados relacionados à aquisição **Arq Dados**, após terem sido exportados e tratados no software Excel contendo 442 pontos.

Figura 44 - Resultado da **Arq Dados** representado no software Excel. Eixo Y= *offset* mais massa (gramas).



Fonte: Elaborados pelo autor.

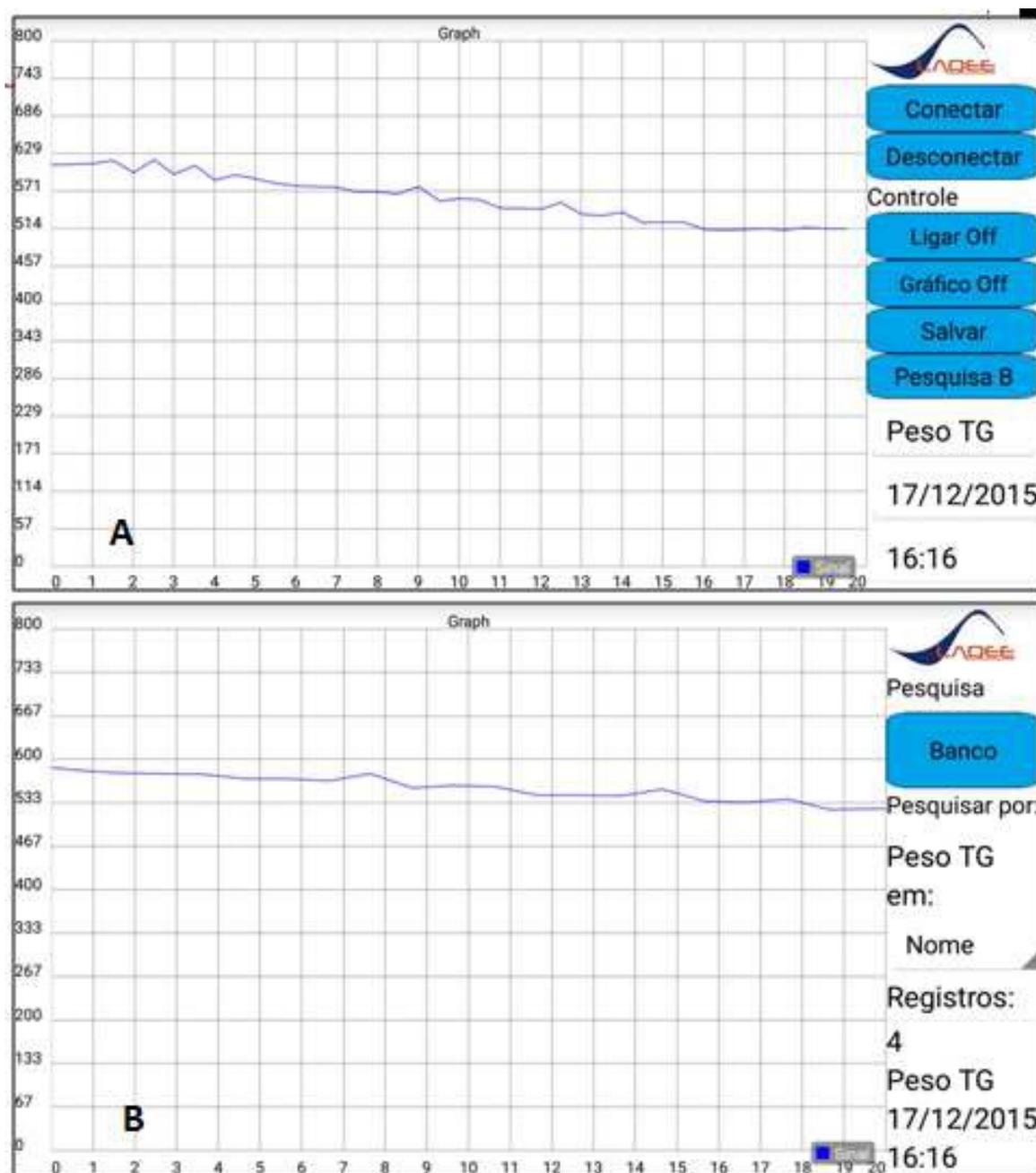
### 7.3.3 Ensaios Utilizando Pesos Relacionados à Tabela 7

Os resultados aqui apresentados são provenientes da vibração que ocorre na barra de pesagem no momento da carga e descarga dos pesos (gramas), conforme nela são colocados, como demonstrado na Tabela 7, ou seja, com tempo de inserção e remoção determinado em três segundos para que fosse possível visualizar a disparidade dos valores apresentados tanto em gráfico como em texto. Assim, foram realizados ensaios, em que se armazenam os respectivos resultados da aquisição no banco de dados e exportam-se os valores para memória de armazenamento do *smartphone*.

Na Figura 45, apresenta-se o resultado da aquisição de dados referente os pesos (gramas) expostos na Tabela 7, de modo que:

- Na tela **A**, apresenta-se a aquisição de dados com seus respectivos valores, nome do arquivo, data e hora do ensaio;
- Na tela **B**, apresenta-se a pesquisa realizada no banco de dados referente à aquisição, contendo os valores e informações básicas da aquisição.

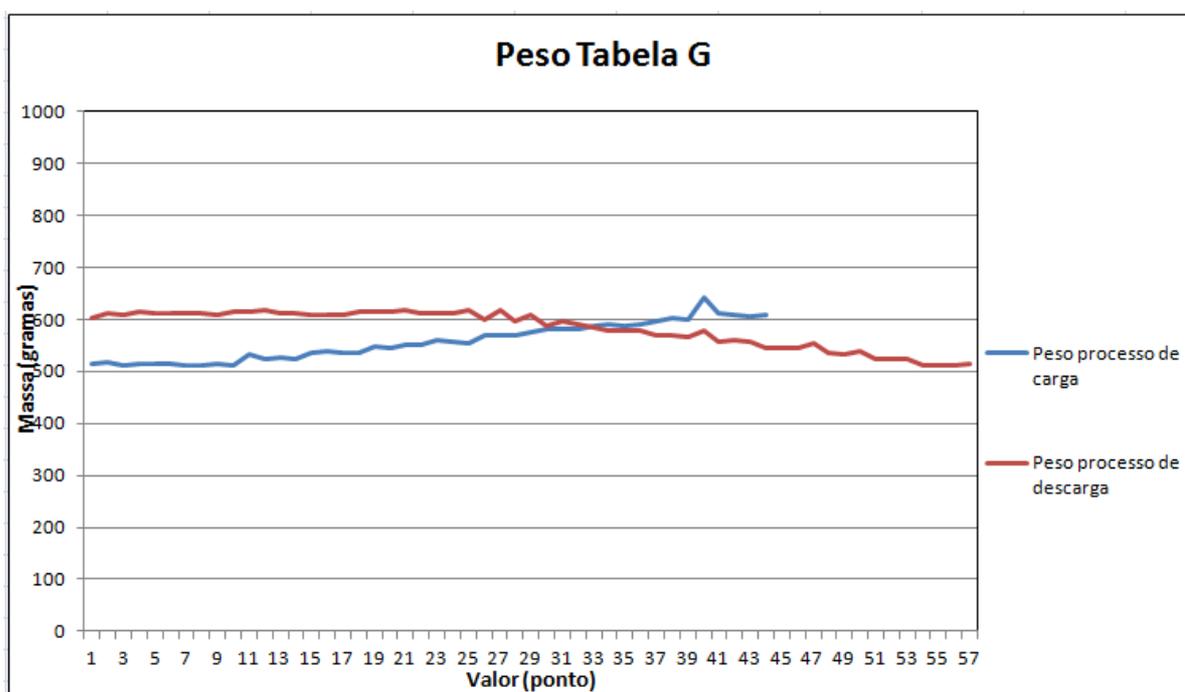
Figura 45 - Resultado gráfico do **Peso TG**: **A**- aquisição de dados; **B**- pesquisa no banco.



Fonte: Elaborado pelo autor.

Ao salvar os dados referente à aquisição **Peso TG** no banco de dados *SQLite*, gerou-se, também, o arquivo de texto, possibilitando o encaminhamento dos dados para e-mail ou para uma base de dados. Sendo assim, o arquivo de texto foi exportado para o software Excel, contexto em que os valores referentes à aquisição pesos de carga e descarga foram tratados. Ou seja, os pesos foram colocados na barra de pesagem sobrepostos até chegar no último valor dos **massas adicionadas** referente à Tabela 7. Em seguida, foram retirados um por vez, respeitando o tempo de três segundos. Tal resultado pode ser visto na Figura 46 com os valores em carga e descarga referente a este ensaio.

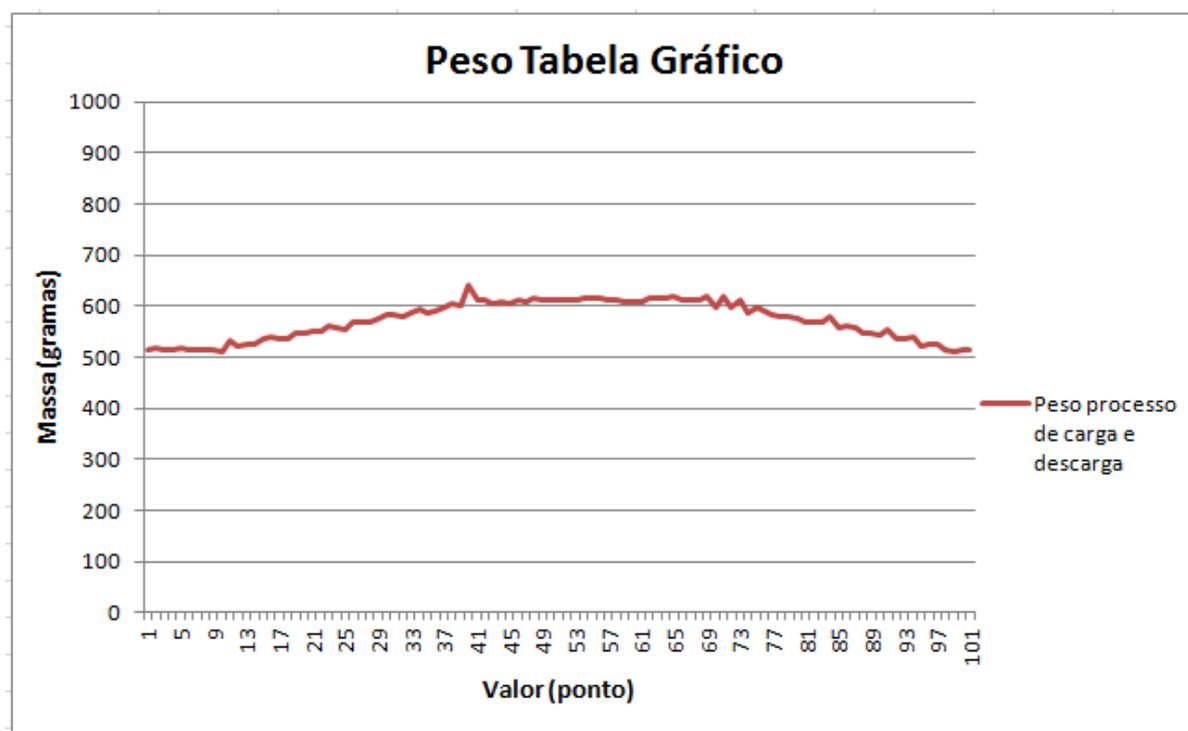
Figura 46 - Resultado em carga e descarga dos valores apresentado na Tabela 7. Eixo Y= *offset* mais massa (gramas).



Fonte: Elaborado pelo autor.

Tratando o arquivo de texto no software Excel, mas de maneira contínua, utilizando o resultado completo da aquisição dos dados denominada de **Peso TG**, podem-se observar, na Figura 47 os dados através de pontos gráficos dos pesos de carga e descarga.

Figura 47 - Resultado gráfico dos valores de carga e descarga. Eixo Y= *offset* mais massa (gramas).



Fonte: Elaborado pelo autor.

Na Figura 48 apresenta-se as informações dos dados salvo na memória do smartphone no formato de arquivo de texto, tal que a tela **A** representa os valores com pesos em carga, e a tela **B** os valores com peso de descarga.

Figura 48 – Dados dos Arquivo de texto salvo na memória de armazenamento do *smartphone*: **A**- arquivo do peso de carga; **B**- arquivo do peso de descarga.



Fonte: Elaborado pelo autor.

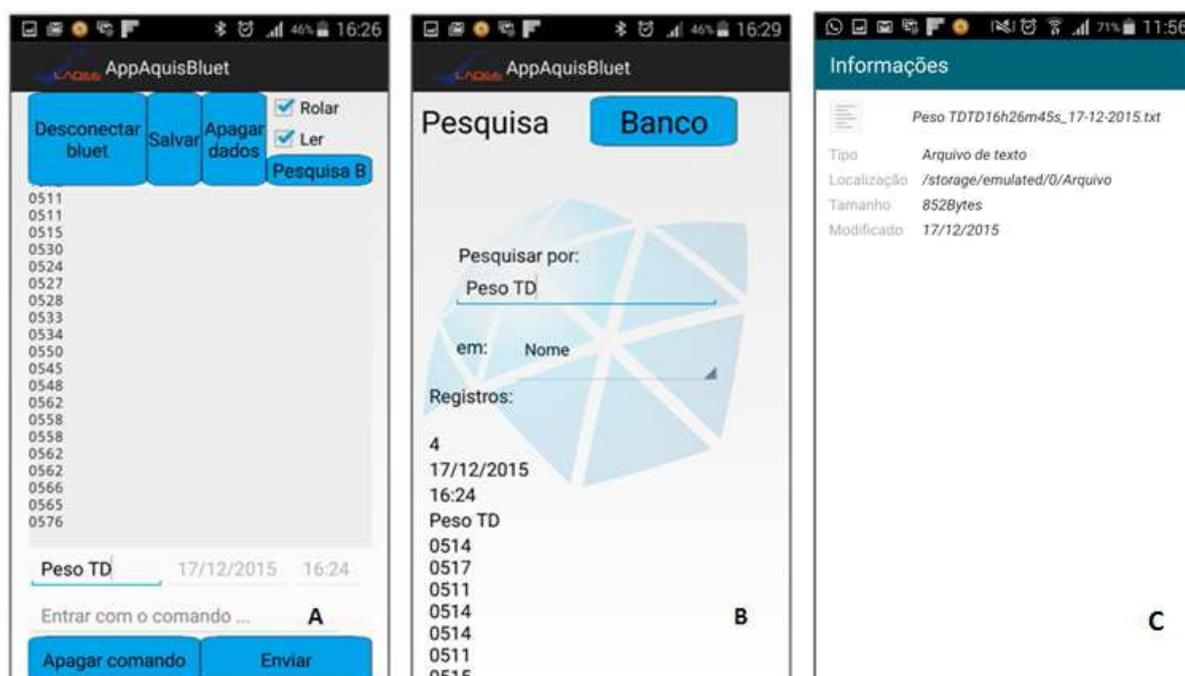
### 7.3.4 Ensaio Utilizando peso Relacionado à Tabela 7 no Formato Texto

Ao realizar aquisições de dados utilizando o condicionador de sinal via *bluetooth* e o aplicativo AppAquisBluet, chegou-se aos resultados, conforme apresentado nos tópicos anteriores. Os resultados processuais são apresentados em gráfico, mas também em formato texto e exportados para a memória do *smartphone*, disponibilizados como imagens, extraídas tanto do aplicativo como do software Excel.

Na Figura 49 apresenta-se o resultado da aquisição dos dados do tipo texto realizada em tempo real. Tal resultado pode ser constatado na tela **A**, contendo os valores dos dados, nome do arquivo, data e hora da aquisição que foi salva no banco de dados *SQLite* e exportado como arquivo de texto para a memória de armazenamento do *smartphone*.

Na tela **B**, observa-se o resultado da pesquisa em banco relacionada à aquisição **Peso TD** contendo os valores dos dados, nome do arquivo, data e hora da realizada aquisição e, a tela **C**, constam informações relacionadas ao arquivo de texto gerado ao exportar os dados **Peso TD** para o *smartphone*.

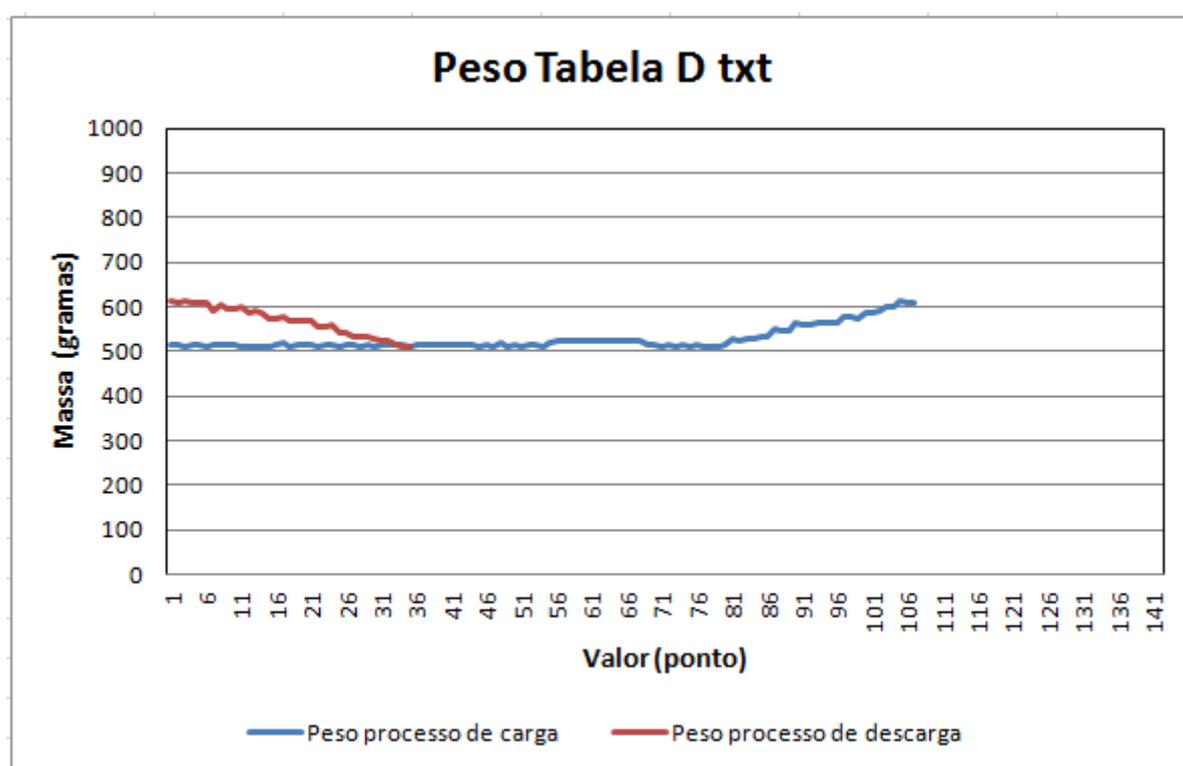
Figura 49 - Arquivo do tipo texto apresentado pelo aplicativo AppAquisBluet: **A**- aquisição de dados; **B**- pesquisa no banco; **C**- informação do arquivo de texto.



Fonte: Elaborado pelo autor.

Seguiu-se o processo de inserção e remoção dos pesos (gramas) na barra de pesagem, com tempo de um segundo entre os valores recebido e três segundos de intervalo entre os novos pesos inseridos e removidos obedecendo à sequência apresentada na Tabela 7 na coluna **massas adicionadas**. Na Figura 50, apresenta-se o resultado da aquisição denominada de **Peso TD** com os respectivos valores recebidos na forma de carga e descarga tratados no software Excel, cujo o resultado é a soma do *offset* com o peso recebido.

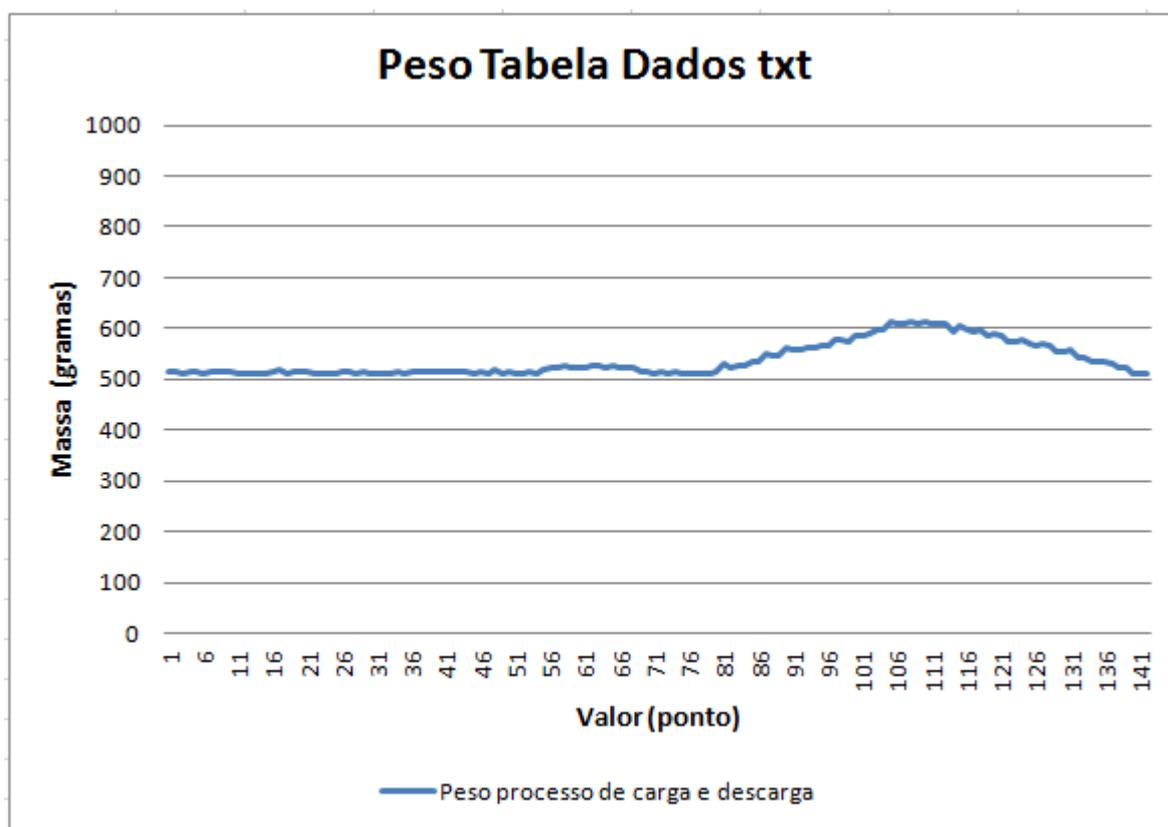
Figura 50 - Arquivo do tipo texto relacionado ao dado **Peso TD**. Eixo Y= *offset* mais massa (gramas).



Fonte: Elaborado pelo autor.

Na Figura 51, apresenta-se o resultado da aquisição **Peso TD** contendo todos os valores recebidos e salvo no banco, utilizando as funcionalidades do aplicativo AppAquisBluet, de modo que, ao salvar tal aquisição em banco, gerou-se, também, o arquivo de texto na memória de armazenamento do *smartphone* para que fosse possível realizar os devidos tratamentos dos dados e apresentar os 141 pontos, utilizando o software Excel.

Figura 51 - Arquivo do tipo texto relacionado ao dado **Peso TD** de forma contínua. Eixo Y= *offset* mais massa (gramas).



Fonte: Elaborado pelo autor.

#### 7.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo, apresentou-se a continuidade dos ensaios envolvendo aquisição de dados em tempo real, utilizando o aplicativo AppAquisBluet juntamente com o hardware condicionador de sinais via *bluetooth*. Contudo, buscou-se demonstrar tanto a potencialidade da aplicação como a tecnologia móvel utilizada no projeto, destacando, a primeiramente, a ponte de Wheatstone em suas características e funcionalidades para este estudo.

Em seguida, apresentaram-se os resultados dos ensaios através de imagens, onde as mesmas demonstraram os resultados da aquisição tanto no aplicativo em formato de pontos gráficos ou em dados texto, além de demonstrar tais resultados, utilizando o software Excel, com objetivo de possibilitar melhor visibilidade nos resultados dos dados tanto no formato de carga como descarga, utilizando a sequência de inserção e remoção dos **massas adicionadas** referente à Tabela 7.

## 8 CONCLUSÕES

Com a tecnologia digital em constante evolução, muitas coisas são realizadas com o auxílio de computadores ou dispositivos móveis. Criando assim as subáreas da computação móvel, redes de comunicação e as implementações de sistemas em dispositivos móveis. Com investimentos das grandes empresas voltados para a tecnologia móvel, hoje, existem *smartphones* de várias marcas e modelos e sistemas operacionais, permitindo ao usuário executar variados serviços, a qualquer hora, em qualquer lugar.

Diante deste cenário, a linha de pesquisa deste estudo voltou-se para a tecnologia móvel. Assim, após feitas várias análises em materiais disponibilizados na literatura vigente, decidiu-se adotar a plataforma de desenvolvimento Android por ser uma plataforma de considerável custo-benefício, quando comparada às demais do mercado, a exemplo da plataforma Apple iOS.

Escolher tal caminho foi simples, visto que a Google, juntamente com seus aliados, disponibiliza para o desenvolvedor, corporativo ou convencional, uma plataforma moderna, flexível e *open-source*, baseada no *kernel* do Linux, além de disponibilizar em seu núcleo uma SDK repleta de ferramentas para auxiliar o desenvolvedor na criação de aplicações cada vez mais sofisticadas.

Outro ponto que contribuiu para utilização da tecnologia Android deve-se ao fato do Android ser líder absoluto no mercado móvel, provavelmente por ser o sistema mais disseminado no mundo não só de *smartphones*, mas de variados dispositivos executados pelo sistema operacional, e possuir a relação custo/benefício mais acessível às variadas classes da sociedade, dados os investimentos realizados pela Google e seus aliados em sua plataforma Android, possibilitando, desse modo, a execução de sua tecnologia tanto em aparelhos mais simples até os mais sofisticados.

Nesse cenário, colocou-se em prática o projeto do AppAquisBluet, utilizando técnicas e características próprias da plataforma Android, chegando ao objetivo proposto para o projeto em questão. Para tanto, apresentou-se o aplicativo AppAquisBluet, desenvolvido na IDE Eclipse, com tecnologia da programação Android, API *bluetooth*, comunicação serial simulada por RF-COMM, implementação gráfica administrada pela biblioteca *GraphView* e métodos de banco de dados *SQLite*, adequados para dispositivos móveis, o que levou ao resultado de aquisições de dados preliminares, utilizando o *notebook* para simular o sistema real, como pode ser observado no Capítulo 5 deste trabalho.

Nesse sentido, demonstraram-se as principais características e funcionalidade da aplicação, apresentando resultados de aquisições de dados em tempo real, tanto como arquivo de texto

ou graficamente no visor da aplicação, haja vista que os resultados podem ser armazenados em banco de dados para futuras análises.

Para emular a transmissão dos dados, foi utilizado o programa de terminal OC-Console, cuja tarefa é simular uma porta serial, para receber e enviar dados, entre o módulo *bluetooth*, e o dispositivo intermediada pela classe *BluetoothSocket*.

Quanto aos dados enviados pelo programa de terminal, foram recebidos e apresentados no visor da aplicação, à medida que eram implementados e tratados, porém, sem alterar suas características.

As funções estabelecidas no aplicativo AppAquisBluet funcionaram perfeitamente como era esperado, ou seja, respeitando todos os parâmetros estabelecidos e implementados em código fonte, seja na navegação de telas, aquisição de dados no formato de arquivo de texto ou graficamente e no armazenamento desses dados em banco.

Para finalizar os testes, utilizou-se o aplicativo AppAquisBluet juntamente com o hardware condicionador de sinais via *bluetooth* constituído pela barra de pesagem, na qual são colocados os pesos (gramas) e seus valores transmitidos através da ponte de Wheatstone para o hardware condicionador de sinal via *bluetooth*. Recebidos em formato analógico, os dados são convertidos para valores digitais e transmitidos via comunicação *Bluetooth* para o aplicativo AppAquisBluet, a fim de serem apresentados no visor da aplicação, o que leva à conclusão tanto da potencialidade da aplicação como da tecnologia móvel utilizada para este projeto.

Os resultados dos ensaios também foram apresentados em imagens, a fim de demonstrar os resultados da aquisição tanto no aplicativo em formato de pontos gráficos ou em arquivo de texto, além de demonstrar tais resultados utilizando o software Excel cujo objetivo foi possibilitar melhor visibilidade dos dados no modo de carga ou descarga pela sequência de inserção e remoção dos **massas adicionadas** referente à Tabela 7.

O projeto desenvolvido e testado para este trabalho não se limita apenas aos ensaios envolvendo a medição de peso, já que pode ser utilizado na medição de resistência, medição de temperatura, medição de pressão e medição de torque, bastando apenas readequar o condicionador de sinal via *bluetooth* e refazer a calibração e os ajustes necessários para que os dados possam trafegar, utilizando a comunicação *bluetooth*, apresentando tais valores no visor do aplicativo AppAquisBluet.

A contribuição científica no desenvolvimento do aplicativo AppAquisBluet reside no fato do usuário poder realizar aquisição de dados em tempo real e armazenar tais dados sem necessidade de recorrer aos métodos tradicionais. Há exemplos, pode-se citar a medição de

torque de um determinado eixo de rotação, de modo que o condicionador de sinal via *bluetooth* esteja acoplado, diretamente, no eixo de rotação transmitindo os ensaios em tempo real. Além disso, é preciso registrar a segurança alcançada na implementação deste projeto, ou seja, ambiente de trabalho seguro, saudável e livre de infortúnios, o que pode se traduzir em conforto e segurança, também, para o usuário do aplicativo AppAquisBluet, visto que seus dados são mantidos em sigilo e segurança.

Como trabalhos futuros pode-se realizar variadas implementações, tanto na interface do aplicativo, como no direcionamento de variados tipos de aquisições de dados como já foi mencionado neste trabalho, bastando realizar alguns ajustes e calibragem do sistema. Quanto ao melhoramento do aplicativo, pode-se implementar o mesmo para ser executado em um *tablet*, pois o mesmo disponibiliza uma tela maior que fornece mais visibilidade e clareza nos resultados dos ensaios aqui mencionados.

Pode-se ainda, fazer outros melhoramentos como a implementação de um cursor no visor gráfico do aplicativo utilizando uma linha na vertical que ao ser acionada tanto para esquerda ou direita mostrará para o usuário o valor correspondente plotado no gráfico. Além da criação de um botão que faz a captura da imagem da aquisição de dados em tempo real e outro botão que tenha a função de gerar um vídeo da aquisição ao ser acionado.

Quanto as cores utilizadas no visor de plotagem tanto dos dados gráficos e do tipo texto, pode-se implementar um botão de ajuste de cores do visor objetivando economia relacionada a bateria, pois quanto mais claro o visor mais consumo de energia terá a bateria. Essas são algumas melhorias que possam ser implementadas nos trabalhos futuros.

## REFERÊNCIAS

- ABREU, L. A mordida da maçã: o projeto da Apple provocante. *Revista Advérbio*, v. 5, n. 09, 2012.
- ANDOLFATO, R. P.; CAMACHO, J. S.; BRITO, G. d. *Extensometria básica*. Ilha Soleira: Núcleo de Ensino e Pesquisa da Alvenaria Estrutural-Universidade Estadual de São Paulo, 2004.
- ANDROID. *A história do Android*. [S. l.], 2015. Disponível em: <<http://www.android.com/intl/pt-BR/br/history/>>. Acesso em: 23 ago. 2015.
- ANDROID, D. *Bluetooth*. [S. l.], 2015. Disponível em: <<http://developer.android.com/guide/topics/connectivity/bluetooth.html#ManagingAConnection>>. Acesso em: 23 maio 2015.
- APPLE. *A experiência móvel mais avançada avançou ainda mais*. [S. l.], 2015. Disponível em: <<http://www.apple.com/br/ios/>>. Acesso em: 23 ago. 2015.
- BECK, R. et al. *Identificação experimental de curvas tensão-deformação a taxa constante por ensaios axiais de impacto em corpos de prova de pvc e pp*. Florianópolis: [s. n.], 2009.
- BEZERRA, E. *Princípios de análise e projeto de sistemas com UML*. 3. ed. Rio de Janeiro: Elsevier, 2015. 416 p.
- BLUETOOTH. *A look at the basics of bluetooth technology*. [S. l.], 2015. Disponível em: <<http://www.bluetooth.com/Pages/Basics.aspx>>. Acesso em: 14 maio 2015.
- BRESIL, R. *Integração sem fio entre redes CRP SEM (WLANs) e redes de sistemas celulares*. São Paulo: Unicamp, 2004. Disponível em: <<http://www.bibliotecadigital.unicamp.br/document/?code=vtls000345131>>. Acesso em: 16 maio 2015.
- CALIL, A. *O que tornou o tablet da HP um grande fracasso*. [S. l.], 2012. Disponível em: <<http://economia.estadao.com.br/noticias/-negocios,o-que-tornou-o-tablet-da-hp-um-grande-fracasso,98032e>>. Acesso em: 05 maio 2015.
- CARVALHO, R. *Empresa retoma o projeto MeeGo e lança novo OS Mobile*. [S. l.], 2012. Disponível em: <<http://tiqx.blogspot.com.br/2012/12/empresa-retoma-o-projeto-meego-e-lanca.html>>. Acesso em: 11 abr. 2015.
- CCM. *Redes sem fio: wireless networks*. 2015. Disponível em: <<http://br.ccm.net/contents/819-redes-sem-fio-wireless-networks>>. Acesso em: 05 ago. 2015.
- CHLAMTAC, I.; CONTI, M.; LIU, J. J.-N. Mobile ad hoc networking: imperatives and challenges. *Ad hoc networks*, Elsevier, v. 1, n. 1, p. 13–64, 2003.
- CONTI, M.; GIORDANO, S. Mobile ad hoc networking: milestones, challenges, and new research directions. *Communications Magazine, IEEE*, v. 52, n. 1, p. 85–96, 2014.
- CORRÊA, U.; PINTO, A.; CODAS, A.; FERREIRA, D.; MONTEZ, C. *Redes locais sem fio: conceitos e aplicações*. Passo Fundo: IV Escola Regional de Redes de Computadores, 2006.

DATE, C. J. *Introdução a sistemas de bancos de dados*. São Paulo - SP: Elsevier Brasil, 2004

GOMES, P. *Comunicação via satélite*: [S. 1], 2015. Disponível em: <<http://pgomes.com.br/arquivos/75febcb272a8e9d760799bf484361fa9.pdf>>. Acesso em: 26 mar. 2015.

Goadrich, M. H. and Rogers, M. P. (2011). Smart smartphone development: iOS versus android. In Proceedings of the 42nd ACM technical symposium on Computer science education, SIGCSE '11, pages 607–612, New York, NY, USA. ACM.

GOZALVEZ, J. South korea launches LTE-advanced [mobile radio]. *Vehicular Technology Magazine*, New York, v. 9, n. 1, p. 10–27, 2014. ISSN 1556-6072.

GRAPVIEW, A. *Key features*. [S. 1.], 2015. Disponível em: <<http://www.android-graphview.org/>>. Acesso em: 23 maio 2015.

GUADAGNINI, P. H.; ROCHA, F. S. da; ELISABETH, V. Projeto de um sensor eletrônico baseado em extensometria para medição de força. *Latin-American Journal of Physics Education*, v. 5, p. 753–762, 2011.

GUIMARÃES, G. *A história do sistema operacional Android*. 2013. Disponível em: <[http://www.dsc.ufcg.edu.br/pet/jornal/agosto2013/materias/historia\\_da\\_computacao.html](http://www.dsc.ufcg.edu.br/pet/jornal/agosto2013/materias/historia_da_computacao.html)>. Acesso em: 14 maio. de 2015.

HAMANN, R. *O líder absoluto no mercado*. [S. 1.], 2014. Disponível em: <<http://www.tecmundo.com.br/sistema-operacional-/60596-ios-android-windows-phone-numericos-gigantes-comparados-infografico.htm>>. Acesso em: 23 maio 2015.

HUBBARD, G. Cooke and Wheatstone: and the invention of the electric telegraph. *Taylor and Francis*, Hoboken, v. 16, p. 179, 2013.

KAUR, P.; SHARMA, S. Google Android a mobile platform: a review. In: ENGINEERING AND COMPUTATIONAL SCIENCES (RAECS), 2014, Gharuan. *Proceedings...* New York: IEEE, 2015. p. 1–5.

LECHETA, R. R. *Google Android: aprenda a criar aplicações para dispositivos móveis com o Android SDK*. 3<sup>a</sup> ed. São Paulo: Novatec, 2013. 822 p.

MACIEJEWSKY, J. *A evolução do Windows CE até Windows Phone 7*. [S. 1.: s. n.], 2011. Disponível em: <<http://blog.maciejewsky.net/blog/post/2011/04/12/-A-evolucao-do-Windows-CE-ate-Windows-Phone-7.aspx>>. Acesso em: 27 mar. 2014.

MASCARENHAS, M.; MARTINS, M.; BULCAO, L.; BRITO, J. de; VIEIRA, V.; DURAN, A. *Um estudo de caso com análise comparativa entre plataformas para aplicações móveis aberta e proprietária: Android e iOS*. Recife: UFPE, 2013. Disponível em: <[www.cin.ufpe.br/~ubibus/artigos/112186.pdf](http://www.cin.ufpe.br/~ubibus/artigos/112186.pdf)>. Acesso em: 21 jul. 2015.

MATEUS, G. R.; LOUREIRO, A. A. F. *Introdução à computação móvel*. Rio de Janeiro: DCC/IM, COPPE/UFRJ, 1998.

MEDNIEKS, Z.; DORNIN, G. L.; NAKAMURA, M. *Programando o Android*. São Paulo: Novatec, 2012. 561 p.

MESSIAS, A. R. *Adicionando um dispositivo serial RS232 com XBee Wi-Fi (IEEE 802.11 b/g/n), numa rede de infraestrutura (BSS - Basic Service Set)*. [S. l.: s. n.], 2015. Disponível em: <<http://www.rogercom.com/XBeeWi-Fi.htm>>. Acesso em: 05 ago. 2015.

MITCHELL, B. *Wireless Standards 802.11a, 802.11b/g/n, and 802.11ac*. [S. l.: s. n.], 2015. Disponível em: <<http://compnetworking.about.com/cs/wireless80211/a/aa80211standard.htm>>. Acesso em: 21 mar. de 2015.

MOREIRA, E. *MWC 2014/ BlackBerry Z3, primeiro smartphone fruto da parceria com a Foxconn, é anunciado*. [S. l.: s. n.], 2014. Disponível em: <<http://targethd.net/-mwc-2014-blackberry-z3-primeiro-smartphone-fruto-da-parceria-com-foxconn-e-anunciado-/>>. Acesso em: 21 abr. 2015.

NIKOLOFSKI, D. R. F. *A quarta geração das redes sem fio: benefícios e evolução*. Curitiba: [s. n.], 2013.

OLIVEIRA, R. N. *Wireless torque transducer*. 2015. 3 p.

PARADA, A. G.; BRISOLARA, L. B. de. A model driven approach for Android applications development. In: COMPUTING SYSTEM ENGINEERING (SBESC), 2012 BRAZILIAN SYMPOSIUM ON, 2012, Brasília. *Proceedings...* New York: IEEE, 2012. p. 192–197.

PAULO, J. V. d. *Desenvolvimento de um aplicativo Android e de uma interface Bluetooth para um dinamômetro biomédico*. 2014. 93 F. Dissertação (Mestrado) – Faculdade de Engenharia, Universidade Estadual Paulista, Ilha Solteira, 2014.

PELLANDA, E. C. *Internet móvel: novas relações na cibercultura derivadas da mobilidade na comunicação*. 2005. 194 p. Tese (Doutorado em Comunicação Social) - Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2005.

PELLANDA, E. C. Comunicação móvel: das potencialidades aos usos e aplicações. *Em questão*, v. 15, n. 1, p. 89-98, 2009.

PEREIRA, L. C. O.; SILVA, M. L. D. *Android para desenvolvedores*. Rio de Janeiro Rio de Janeiro: Brasport, 2009.

PEREIRA, L. C. O.; SILVA, M. L. D. *ANDROID para desenvolvedores*. 2. ed. Rio de Janeiro: Brasport, 2012.

QUERINO, L. C. F. *Desenvolvendo seu primeiro aplicativo Android: entre de cabeça no mundo dos aplicativos móveis, criando e publicando seu próprio programa para o sistema líder do mercado!* São Paulo: Novatec, 2013. 247 p.

RATTMANN, A. C.; RATTMANN, R. P. *PCS - Sistemas de Comunicação Pessoal: jornal técnico*. [S. l.: s. n.], 2014. Disponível em: <<http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=474>>. Acesso em: 25 mar. 2015.

REUTERS. *Samsung executive says Galaxy S5 to outsell S4, sees second quarter rollout for Tizen phone*. [S. l.], 2014. Disponível em: <<http://www.reuters.com/article/2014/04/16-/us-samsung-elec-sales-idUSBREA3F09320140416>>. Acesso em: 24 abr. 2015.

REZA, H.; MAZUMDER, N. A secure software architecture for mobile computing. In: INFORMATION TECHNOLOGY: NEW GENERATIONS (ITNG), 2012 NINTH INTERNATIONAL CONFERENCE ON, 2012, Grand Forks, *Proceedings...* New York: IEEE, 2012. p. 566–571.

RIBEIRO F. I. N., FERRAZ, F. S. Uma abordagem comparativa do desenvolvimento de aplicações para dispositivos móveis. [S. l.: s. n.], 2011.

Tracy, K. (2012). Mobile application development experiences on Apple's ios and Android os. *Potentials, IEEE*, 31(4):30–34.

SAMPAIO, C. A. d. P.; PASSOS, E. F.; DIAS, G. P.; CORREA, P. C. Desenvolvimento e avaliação de um anemômetro de fio quente operando à temperatura constante. *Revista Brasileira de Engenharia Agrícola e Ambiental*, Campina grande, v. 2, n. 2, p. 229–234, 1998.

SEBRAE. *Quarta geração de celulares nas cidades-sede da Copa de 2014*. São Paulo, 2014. Disponível em: <http://www.boletimdoempreendedor.com.br/2014/boletim-.aspx?codBoletim=5>>. Acesso em: 25 mar. 2015.

SILVA, L. A. *Apostila de Android: programando passo a passo : programação Básica 6ª ed*. Rio de Janeiro, 2013. Disponível em: <[www.apostilaandroid.net](http://www.apostilaandroid.net)>. Acesso em: 25 mar. 2015.

SMITH, D.; FRIESEN, J. *Receita Android: uma abordagem para resolução de problemas*. Rio de Janeiro: Ciência Moderna, 2012. 473 p.

SQLITE. *About SQLite*. [S. l], 2015. Disponível em: <[www.sqlite.org/about.html](http://www.sqlite.org/about.html)>. Acesso em: 25 mar. 2015.

THOMPSON, T. J.; KUMAR, C. B.; KLINE, P. J. *Bluetooth application programming with the Java APIs essentials edition*. San Francisco: Morgan Kaufmann, 2008.

VIEIRA, V.; FIALHO, A.; MARTINEZ, V.; BRITO, J.; BRITO, L.; DURAN, A. An exploratory study on the use of collaborative riding based on gamification as a support to public transportation. In: COLLABORATIVE SYSTEMS (SBSC), 2012 BRAZILIAN SYMPOSIUM ON, 2012, [S. l.]. *Proceedings...* [S.l.: s.n.], 2012. p. 84–93.

VIEIRA, V.; SALGADO, A. C.; TEDESCO, P.; TIMES, V.; FERRAZ, C.; HUZITA, E.; CHAVES, A. P.; STEINMACHER, I. The ubibus project: using context and ubiquitous computing to build advanced public transportation systems to support bus passengers. In: SIMPÓSIO BRASILEIRO DE SISTEMAS DE INFORMAÇÃO, 8., 2012, São Paulo. *Anais...* São Paulo: SBSI, 2012. p. 55-60.

ZAPATA, B. C.; NIÑIROLA, A. H.; IDRI, A.; FERNÁNDEZ-ALEMÁN, J. L.; TOVAL, A. Mobile phrs compliance with Android and iOS usability guidelines. *Journal of Medical Systems*, New York, v. 38, n. 8, p. 1–16, 2014.