UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Desenvolvimento de dispositivo sem fio para medição de grandezas elétricas para sistemas *Google Android*

Aluna: Priscila dos Reis
Professor Orientador: Evandro Luis Linhari Rodrigues

SÃO CARLOS
2014

PRISCILA DOS REIS

Desenvolvimento de dispositivo sem fio para medição de grandezas elétricas para sistemas *Google Android*

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo.

Curso de Engenharia Elétrica com ênfase em Eletrônica

Professor Orientador: Evandro L. L. Rodrigues

SÃO CARLOS

2014

Reis, Priscila dos
R375d Desenvolvimento de dispositivo sem fio para medição de grandezas elétricas para sistemas Google Android / Priscila dos Reis; orientador Evandro Luis Linhari Rodrigues. São Carlos, 2014.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2014.

1. Instrumentação. 2. Multímetro. 3. Microcontrolador. 4. Bluetooth. 5. Google Android. 6. Phonegap. I. Título.

FOLHA DE APROVAÇÃO

Nome: Priscila dos Reis

Título: "Desenvolvimento de dispositivo sem fio para medição de grandezas elétricas para sistemas Google Android"

Trabalho de Conclusão de Curso defendido e aprovado em 25 / 06 / 2014,

com NOTA 9,9 (Nove, Nove), **pela Comissão Julgadora:**

Prof. Associado Evandro Luís Linhari Rodrigues - (Orientador - SEL/EESC/USP)

Prof. Associado Ivan Nunes da Silva - (SEL/EESC/USP)

Dr. Danilo Hernane Spatti - (Pós-Doutorado -SEL/EESC/USP)

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Homero Schiabel

À minha família, especialmente aos meus pais, irmãos e tios, por todo o amor, apoio e incentivo.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

ADC	*Analog to Digital Converter*
ADK	*Accessory Development Kit*
API	*Application Programming Interface*
BLE	*Bluetooth Low Energy*
CAD	*Computer-Aided Design*
CSS	*Cascading Style Sheets*
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
HTML	*HyperText Markup Language*
ICSP	*In-circuit Serial Programming*
IDE	*Integrated Development Environment*
PCB	*Printed Circuit Board*
PDA	*Personal Digital Assistant*
RAM	*Random Access Memory*
SDK	*Software Development Kit*
UART	*Universal Asynchronous Receiver/Transmitter*

Resumo

Esse projeto visou à construção de um dispositivo de baixo custo para a aquisição das grandezas elétricas tensão, corrente e resistência que, por meio de comunicação sem fio sob o protocolo *Bluetooth*, envia a um *smartphone* com sistema *Google Android* os dados adquiridos. Os dados são recebidos por um aplicativo *Android* desenvolvido para o trabalho, exibidos na tela do *smartphone* ao usuário e armazenados para acesso futuro. O aplicativo também processa as informações que recebe, apresentando os valores de mínimo e máximo lidos, além de apresentar em gráfico os valores adquiridos. Os resultados apresentados revelaram que o circuito projetado, comparado a um instrumento comercial, realizou as medições das grandezas com níveis de diferença inferiores a 5%, o que permite concluir que o trabalho atingiu completamente os objetivos propostos com destaque para o baixo custo dos componentes utilizados.

Palavras-chave: Instrumentação, Multímetro, Microcontrolador, *Bluetooth*, *Google Android, Phonegap*.

Abstract

This project aimed the development of a low-cost device for measuring the electric voltage, current and resistance that sends the acquired data through wireless communication - using the Bluetooth protocol - to a Google Android smartphone. The data is received by an Android application developed for this project, displayed to the user on the Smartphone and stored for future access. The application also processes the information received, showing the readings of minimum and maximum and presenting graphically the acquired values. The results obtained revealed that circuit developed for this project, if compared to a commercial tool, made the measurements at magnitudes of difference with levels below 5%, indicating that this project fully achieved its goals with the highlight of using low cost components.

Keywords: Instrumentation, Multimeter, Microcontroller, Bluetooth, Google Android, PhoneGap.

Capítulo 1

Introdução

1.1 Introdução

O uso de dispositivos móveis, como *smartphones* e *tablets*, é cada vez mais presente, já que estes dispositivos possuem rápida evolução, tornando-se cada vez mais robustos, baratos, menores, além de apresentarem maior autonomia de bateria, o que permite que uma série de aplicações seja realizada com eles. Dentre os sistemas mais populares disponíveis para este tipo de equipamento estão o sistema *Google Android*, *Apple iOS* e *Microsoft Windows Phone*.

O sistema *Android*, que é baseado em *software* livre, permite, diferente dos outros sistemas para plataformas móveis citados, a criação e publicação de aplicações por meio do usuário de uma forma mais simples. Tal conceito implica na vasta criação por meio da comunidade de usuários de diversas formas de aplicação, e o sistema conta com formas de acesso aos recursos e periféricos do sistema em que é executado.

Em conjunto à criação de aplicações para plataformas móveis, algo no mesmo sentido pode ser observado na área da eletrônica embarcada e aplicações em sistemas de baixo nível. A oferta de soluções de *hardware* que permitem um usuário comum ter contato com conceitos de eletrônica é hoje muito grande, e assim o estudo e desenvolvimento de projetos nesta área são facilitados.

Usando o conceito de que estes aparelhos móveis são telas com um grande poder computacional, torna-se interessante a criação de objetos de *hardware* que incrementem as suas capacidades no sentido de novas possibilidades, não imaginadas pelos seus idealizadores, que pensaram em recursos para um usuário comum, mas que também permitem a comunicação com outros dispositivos, tornando possível a interação entre sistemas de *hardware* fechado e outros tipos de dispositivos, que podem ser criados e atender as necessidades e especificações de um usuário, para a realização de uma função não prevista no uso comum dos dispositivos móveis, mas que se beneficia de suas características.

Equipamentos de medição portáteis são amplamente empregados em laboratórios de pesquisa e ensino, e geralmente utilizam telas de cristal liquido, ou mostradores de ponteiro para a exibição de suas leituras, no entanto seu uso se mostra limitado quando uma quantidade de dados elevada existe, assim, o uso de computadores permite o armazenamento e processamento, ampliando as capacidades destes equipamentos [1].

Visando o estudo de aplicações que expandam as capacidades destes dispositivos com o uso de circuitos eletrônicos, foi proposto um conjunto de aplicação e *hardware* dedicado à aquisição de medições elétricas e exibição em dispositivos que possuam sistema operacional baseado em *Google Android*, permitindo o acesso remoto a medições realizadas por um aparelho utilizando comunicação sem fio entre os equipamentos. O uso deste tipo de comunicação permite ao usuário observar e realizar as medições de forma mais confortável e segura, além disso, o *hardware* de dimensões reduzidas permite ao usuário fácil armazenamento para transporte e uso. A grande motivação do uso de plataformas móveis está em adicionar a um equipamento uma tela, que possui como característica a alta resolução, se comparada a soluções comerciais de medição de baixo custo e mobilidade, tornando possível a criação de interfaces amigáveis e interessantes para um usuário, ampliando suas possibilidades.

1.2 Objetivos

Como objetivos desse trabalho pode-se salientar:

- Construção de um dispositivo de baixo custo para aquisição das medidas elétricas: tensão, corrente e resistência.
- Desenvolvimento e implementação do sistema *Bluetooth* para o microcontrolador do dispositivo, possibilitando a comunicação com o *smartphone*.
- Desenvolvimento de um *software* para o *smartphone* para exibição das leituras recebidas do dispositivo.
- Gravação e armazenamento das medidas elétricas em um arquivo.

1.3 Organização do Trabalho

Este trabalho está dividido em capítulos, sendo eles estruturados da seguinte maneira:

- *Fundamentação Teórica*: descrição dos conceitos relevantes para o projeto, mostrando os tópicos e áreas que o envolvem, assim como as plataformas e sistemas utilizados;
- *Descrição do projeto e Desenvolvimento*: apresenta as etapas, materiais e ferramentas do projeto. Também descreve o dispositivo construído para as medições elétricas e os circuitos utilizados, bem como os programas desenvolvidos tanto para o microcontrolador como para o *smartphone* e as funções e conceitos aplicados.
- *Resultados*: apresenta uma discussão e análise dos resultados obtidos com o *hardware* e *software*, além de mostrar os testes realizados em bancada para validar tanto o programa como o circuito projetado.
- *Conclusão*: considerações finais e validação dos objetivos do projeto.

Capítulo 2

Fundamentação Teórica

Neste capítulo será apresentado um embasamento geral a respeito dos temas principais que englobam esse projeto: Instrumentação Eletrônica, Microcontroladores e Sistemas Operacionais. Também serão mostradas a composição e funcionamento de alguns dos componentes e tecnologias utilizados ao longo do desenvolvimento do trabalho.

2.1 Instrumentação Eletrônica

A área de estudo de instrumentação eletrônica visa à conceituação e desenvolvimento de métodos e técnicas para a mensuração de grandezas elétricas, com a utilização de instrumentos, de forma que não interfiram de forma substancial ao ponto que afetem as medições no circuito de interesse.

Dentre os tipos de instrumentos, existem os analógicos e digitais. Os instrumentos analógicos usam como componente fundamental o galvanômetro, sendo o tipo mais comum conhecido como instrumento de bobina móvel e imã permanente. Este componente funciona com o princípio de que quando uma corrente elétrica circula através da bobina é formado um campo magnético que interage com o campo do imã permanente e uma agulha fixada na bobina móvel é utilizada para indicar em uma escala graduada, sendo essa uma referência para a visualização de medições. Desta forma, pode-se inferir que o galvanômetro pode ser utilizado de forma básica para a medição de corrente elétrica. Para a medição de outras grandezas elétricas são utilizados circuitos auxiliares responsáveis pela conversão destas grandezas em níveis de corrente.

A instrumentação digital utiliza de princípios de conversão de valores de sinais analógicos em valores de palavra digital binária. Este tipo de instrumento permite uma melhor manipulação e visualização das medidas se comparado com o de um instrumento analógico. Enquanto um instrumento analógico sofre perda de precisão

dado erro de leitura de um usuário, o instrumento digital exibe em seu mostrador um valor cuja precisão está diretamente ligada a resolução dos circuitos de amostragem, e das capacidades do *display* mostrador. As capacidades de processamento e armazenamento em memória destes equipamentos permitem manipulações de medições, como por exemplo, o armazenamento de uma leitura, além do processamento de dados, para cálculo de valores máximos e mínimos dentro de uma janela de medições, além de outras possibilidades.

2.1.1 Conversor Analógico-Digital

Com o intuito de se processar de forma computacional os valores medidos é necessário converter os valores de sinais analógicos em valores digitais, para isto deve ser utilizado um circuito especifico para esta aplicação, sendo conhecido como conversor analógico-digital, ou ADC, do inglês *Analog to Digital Converter*.

Dentre os princípios empregados na confecção deste tipo de componente, o mais comum é por meio do uso de circuitos comparadores. Quando o nível de tensão de entrada ultrapassa um nível de tensão de referência ocorre um incremento na palavra digital. Quanto maior for a quantidade de comparadores empregados, maior é a resolução do equipamento, e portanto, mais precisa será a conversão, dados os cuidados no projeto do conversor.

2.1.2 Medição de Tensão

A medição de tensão ocorre a partir da queda de tensão sobre um resistor, denominado *shunt*. Este componente é responsável por desviar o caminho da corrente gerando uma tensão de referência. É desejado que este resistor possua o maior valor possível, de forma a drenar a menor quantidade de corrente do circuito a ser medido, sendo ideal uma resistência infinita.

A conexão do instrumento de medição no circuito de interesse é mostrada na *Figura 1*. Neste caso se deseja medir a tensão sobre o resistor R2.

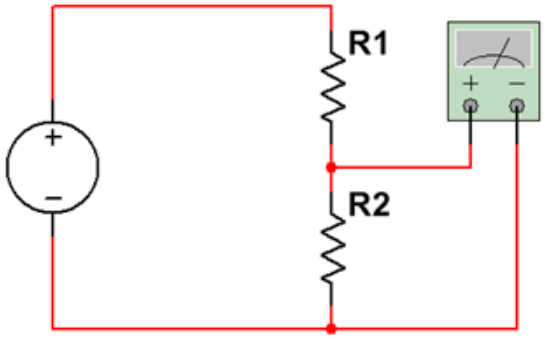


Figura 1 - Medição de Tensão

2.1.3 Medição de Corrente

Para a medição de corrente elétrica, é necessário que o instrumento de medição seja conectado em série ao circuito de interesse. Isto se deve ao fato de que a corrente deve circular pelo instrumento, além deste adicionar a menor resistência possível, sendo idealmente igual a zero, de forma a não afetar a dinâmica do circuito medido.

O diagrama exemplificando o procedimento para medição de corrente elétrica é mostrada na *Figura 2*. Neste caso, deseja-se medir a corrente que circula através do resistor R1, para isto basta ligar o instrumento de medição em série com o resistor R1.

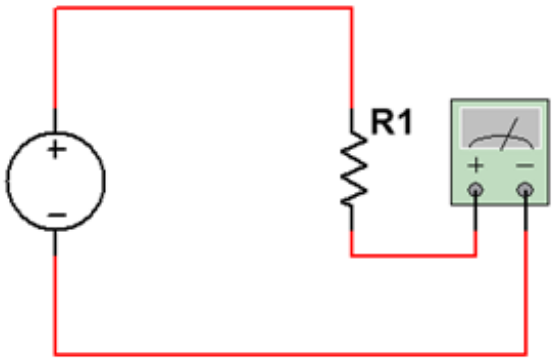


Figura 2 - Medição de Corrente

2.1.4 Medição de Resistência

A medição de resistência ocorre a partir da medição da queda de tensão sobre o elemento de interesse, dado uma referência de tensão e resistência fixa. Desta forma o elemento resistivo que se deseja medir deve estar desconectado para que outros elementos não interfiram na medição.

A medição de resistência por meio de um instrumento é mostrada na *Figura 3*. Basta conectar as pontas de prova do instrumento nos terminais do componente resistor a que se deseje medir.

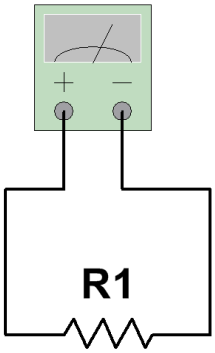


Figura 3 - Medição de Resistência

2.2 Microcontroladores

Um dos grandes meios facilitadores ao desenvolvimento de aplicações robustas em dimensões diminutas são os microcontroladores. Estes componentes são a junção de microprocessador e diversos periféricos, como por exemplos memória RAM (*Random Access Memory*) e memória EEPROM (*Electrically-Erasable Programmable Read-Only Memory*), que permitem que com um pequeno número de componentes se possa executar funções bastante complexas e que também permite a criação de diversas outras aplicações com o incremento de outros dispositivos através de padrões de conexão e protocolos de comunicação.

Os microcontroladores, assim como microprocessadores, executam funções programadas. O programa, escrito por um usuário, é armazenado em uma memória não volátil presente neste dispositivo, e executado quando este está em operação. Podendo

ser escrito em diversas linguagens de programação, estas permitem a abstração de diversos conceitos de baixo nível simplificando assim a criação de aplicações.

Com os padrões de linguagens de programação e sua popularização, é possível também a portabilidade para outras plataformas dado a necessidade do usuário, seja por capacidade de processamento, memória, ou presença de outros periféricos. Dentre estes, estão a comunicação serial UART (*Universal Asynchronous Receiver/Transmitter*) e conversão analógico/digital, utilizadas neste trabalho.

Para o dispositivo desenvolvido, escolheu-se o microontrolador ATmega328P[2] da *Atmel*, presente na plataforma *Arduino*, comentada a seguir.

2.2.1 Plataforma *Arduino*

De acordo com o site oficial [3], o *Arduino* é uma plataforma de prototipagem eletrônica aberta (*open-hardware*), baseada em *hardware* e *software* flexíveis, destinado a qualquer pessoa interessada em criar objetos ou ambientes interativos. Seu microcontrolador é programado com a linguagem de programação *Arduino*, baseada na linguagem *Wiring*, e seu ambiente de desenvolvimento foi escrito em *Java* e baseado no ambiente *Processing* [4].

A plataforma *Arduino* é multiplataforma, funcionando em *Windows*, *Macintoch OSX* e sistemas operacionais *Linux* e pode ser baixado gratuitamente. Já as placas, podem ser adquiridas já montadas ou construídas manualmente, uma vez que o projeto do *hardware* (arquivos de CAD) está disponível sob licença *open-source*, sendo possível adaptá-lo de acordo com as necessidades do usuário.

Além disso, esta plataforma é bastante flexível e permite a compatibilidade do programa escrito para diversos tipos de microcontroladores, possuindo como característica marcante a rápida prototipagem de circuitos embarcados. Desta forma, caso o projeto desenvolvido necessite de mais recursos, a substituição do microcontrolador pouco interfere no programa desenvolvido.

O *Arduino* pode ser usado no desenvolvimento de diversos projetos por permitir o controle de vários dispositivos como sensores, luzes, motores, displays LCD (*Liquid-Crystal Display*), entre outros.

A *Figura 4* mostra um dos modelos de placa, o *Arduino Duemilanove*, baseado no ATmega328P, cujas características são:

- 32KBytes de memória *Flash*;
- 1KByte de memória EEPROM;
- 2KBytes de memória RAM;
- 2 Timers/Counter de 8-bit;
- 1 Timer/Counter de 16-bit;
- 6 canais de PWM;
- Conversor AD (Analógico/Digital) de 6 canais e 10-bit;
- Tensão de operação: 1,8 V a 5,5 V;
- Temperatura de operação: -40º C a 85º C.

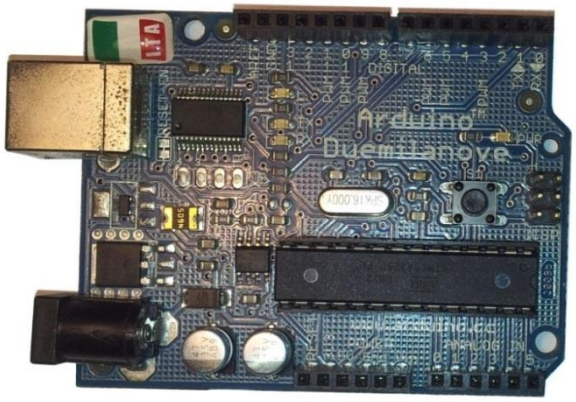


Figura 4 - *Arduino Duemilanove*

2.3 Sistemas Operacionais

O conceito de computação ligada a dispositivos de telefonia é bastante antigo, datando do ano de 1973. No entanto, sua disponibilidade para o público consumidor veio anos depois, assim como seu advento.

Durante a década de 1990 muitos usuários de telefonia móvel utilizavam em conjunto a aparelhos celulares dispositivos chamados de assistentes pessoais, ou PDA, do inglês *Personal Digital Assistant*. Estes equipamentos possuíam recursos de conexão

com internet, reprodução de arquivos de mídia, suporte a leitura e escrita de texto além de uma série de outras aplicações.

No início dos anos 2000 se tornaram comuns dispositivos que combinavam aparelhos de telefonia com PDA, no entanto possuíam recursos limitados. A partir daí, começaram a surgir novos sistemas operacionais móveis que faziam uso de recursos de *hardware* que começaram a surgir nos equipamentos, trazendo uma elevada gama de oportunidade para desenvolvedores na criação de aplicações de diversos tipos.

Assim como em computadores de mesa, nas plataformas móveis o sistema operacional é um programa responsável por gerenciar recursos do sistema, como memória e periféricos, além de gerenciar processos e dados, sendo também uma interface com o usuário.

O principal sistema operacional móvel que trouxe conceitos que são empregados até hoje foi o *Symbian*, sistema que até o final de 2010 era o mais popular dentre os aparelhos s*martphones*.

No ano de 2007, a *Apple* lançou no mercado o *iPhone*, um dos primeiros dispositivos com suporte a tela sensível a múltiplos toques. Este recurso adicionou capacidades de interação a este tipo de dispositivo que mudaram muitos conceitos aplicados em plataformas móveis. Já em 2008, foi lançado o primeiro telefone celular com sistema *Google Android*. Este sistema de código aberto foi um dos grandes responsáveis pela popularização deste tipo de equipamento.

Hoje os três principais sistemas que são presentes nos *smartphones* são o *iOS*, da *Apple*, o *Google Android* e o *Windows Phone*.

O sistema operacional escolhido para esse projeto foi o *Android*, que será comentado a seguir. Os motivos da escolha serão mostrados no *Capítulo 3*.

2.3.1 *Android*

Android é um sistema operacional de código aberto para dispositivos móveis baseado em *Linux*. Foi desenvolvido pela *Open Handset Alliance*, liderada pelo *Google Inc.*

De acordo com o site oficial [5], o *Android* é o sistema usado em mais de um bilhão de *smartphones* e *tablets* em todo o mundo. Com base nas contribuições da

comunidade de código aberto *Linux* e mais de trezentos tipos de *hardware*, *software* e operadoras parceiras, o *Android* se tornou o sistema operacional móvel de mais rápido crescimento.

O primeiro aparelho móvel com esse sistema foi o *HTC Dream* ou G1, lançado oficialmente em 22 de outubro de 2008 nos Estados Unidos. Atualmente, o *Android* está na versão 4.4, chamada de *KitKat*. Uma curiosidade é fato de as versões possuírem sempre o nome de um doce, seguindo a ordem alfabética.

Para o desenvolvimento de aplicativos para a plataforma, é necessário utilizar o *Android SDK* (*Software Development Kit* ou Quite de Desenvolvimento de Programa), que fornece as bibliotecas de API e as ferramentas necessárias para a construção, testes e depuração de aplicativos e pode ser baixado gratuitamente no site oficial para desenvolvedores [6].

Por permitir o acesso a grande quantidade de recursos de *hardware* nos dispositivos *Android* e facilitar a criação da interface com o usuário foram escolhidos os *frameworks Phonegap* e *Fries* para o desenvolvimento do programa nesse sistema operacional. Ambos são descritos a seguir.

2.3.1.1 *PhoneGap*

PhoneGap [7] é um *framework* de código aberto, originalmente desenvolvido pela *Nitobi Software* e, em 2011, adquirido pela *Adobe Systems*. É baseado na plataforma *Apache Cordova* e ambos permitem a criação de aplicativos móveis usando HTML5, *Javascript* e CSS sem a necessidade de usar as linguagens nativas de cada plataforma.

No momento da compilação da aplicação, a API (Interface de Programação de Aplicativos) faz o acesso aos sensores, dados, status de rede, além de uma série de outros periféricos da plataforma alvo, na qual a aplicação irá ser executada, seguindo os padrões de cada uma, simplificando assim a criação de uma mesma aplicação para plataformas diferentes.

Dependendo do dispositivo, possui suporte a recursos como acelerômetro, câmera, contatos, GPS, sistemas de arquivos, notificações (aleta, som, vibração) e disponibilidade de rede.

Atualmente, *PhoneGap* suporta o desenvolvimento para os sistemas operacionais *Amazon Fire OS, Android, BlackBerry, Firefox OS, iOS, Ubuntu, Windows Phone, Windows 8* e *Tizen.*

Os limites do uso do *PhoneGap* podem ser expandidos com o uso de *plugins*, permitindo ao desenvolvedor acesso a outros recursos do aparelho. O *Bluetooth Serial* [8], por exemplo, permite que a aplicação desenvolvida se conecte e comunique com dispositivos utilizando do protocolo de comunicação sem fio *Bluetooth*. É suportado pelas plataformas *Android* e iOS, embora neste segundo seja necessário o uso de módulos do tipo BLE (*Bluetooth Low Energy*).

Já o *plugin SocialSharing* [9] permite o compartilhamento de conteúdos, como textos, imagens, links e arquivos, com o auxílio do *widget* de compartilhamento nativo presente nas plataformas *Android*, iOS, e *Windows Phone*.

Além desses, é possível encontrar uma série de *plugins* desenvolvidos pela comunidade que adicionam recursos que podem ser utilizados na criação de diversas aplicações.

2.3.1.2 Fries

Fries [10] é um *framework* que permite a criação de aplicativos *Android* utilizando as linguagens HTML, CSS e *JavaScript*. Possui vários componentes *Android*, como barras de navegação, botões, listas, formulários e guias. É de código aberto e foi inspirado pelo *Ratchet*, também um *framework* de linguagem *web*, mas com suporte para *Android* e iOS.

Como o *PhoneGap* (descrito anteriormente) permite a criação de aplicações com o uso de linguagens de programação *web*, a interface de usuário *Fries* mostra-se uma eficiente escolha para o desenvolvimento de aplicativos para plataforma em questão, uma vez que possui o mesmo conceito visual.

Capítulo 3

Descrição do Projeto e Desenvolvimento

Neste capítulo, serão apresentados a descrição, os materiais utilizados além de todos os passos e atividades realizadas no projeto em questão.

3.1 Descrição do Projeto

O projeto consiste no desenvolvimento de um dispositivo de aquisição de medidas elétricas (tensão, corrente e resistência) que, por meio da tecnologia de comunicação *Bluetooth* envia os dados adquiridos a um *smartphone* com sistema *Google Android* para serem tratados e exibidos na tela ao usuário em forma de números e gráfico, além de poderem ser armazenados para futuro acesso.

O *hardware* desenvolvido para a realização das medições e comunicação com os dispositivos móveis *Android* é baseado em uma plataforma construída com o uso de um microcontrolador AVR ATmega328P, fabricado pela *Atmel Corporation*. O circuito projetado faz uso de conversores analógico/digitais presentes no microcontrolador, e circuitos atenuadores e condicionadores de sinal. Tendo em vista a necessidade de o dispositivo final ser portátil, ele é alimentado de forma que possibilite isto. Assim, foi utilizado um circuito integrado regulador de tensão para a adequação da tensão das baterias utilizadas para uma tensão compatível com a de alimentação do microcontrolador e demais componentes.

Entre os sistemas operacionais existentes nos dispositivos móveis, como *smartphone* e *tablets*, os mais populares são o *iOS*, presente em aparelhos da fabricante *Apple Inc.*, o *Android*, sistema desenvolvido pela *Google Inc.*, e Windows 8 e *Windows Phone 8*, da *Microsoft Corporation*.

A escolha do sistema *Google Android* foi feita já que esse sistema operacional é bastante utilizado por diversos equipamentos móveis presentes no mercado e popular entre usuários deste tipo de dispositivo e suas aplicações, além da facilidade de obtenção de ferramentas sem custo para desenvolvimento, documentação e exemplos.

Outro recurso existente nesse sistema que favorece sua escolha é o livre acesso a comunicação sem fio utilizando o protocolo *Bluetooth* ou *WiFi*, ou através de conexão USB (*Universal Serial Bus*).

Uma plataforma chamada *Accessory Development Kit* (ADK) é uma implementação de referência para fabricantes de dispositivos físicos e entusiastas em eletrônica para o desenvolvimento de acessórios para equipamentos que utilizem sistema operacional *Google Android*. A criação dessas aplicações é facilitada através do uso do protocolo *Android Open Accessoy* (AOA), que permite a comunicação com dispositivos *Android* por meio de uma conexão *USB* e um *hardware* baseado na plataforma *Arduino*.

Dentre as possibilidades de comunicação, além do uso da interface USB, pode-se também utilizar de protocolos sem fio, como a comunicação *WiFi* 802.11b/g e *Bluetooth*. Ambas permitem o acesso à troca de dados entre dispositivos, necessário para o projeto desenvolvido, e possuem soluções de *hardware* que se conectam a microcontroladores.

Comparando ambos, o uso de conexão *WiFi* necessita da criação de uma rede local com o uso de um recurso nativo do sistema *Android*, desde a versão 2.2, chamado *WiFi tethering hotspot*, que permite a criação de uma rede Ad Hoc P2P entre o dispositivo e a placa. Já para a conexão *Bluetooth* permite a conexão com dispositivos e troca de dados desde que esteja habilitada nas configurações de rede sem fio, e o pareamento entre equipamentos efetuado.

Como vantagem do uso do *Bluetooth*, têm-se um menor consumo de energia e módulos de *hardware* para criação de aplicações embarcadas de custo mais baixo, tendo como desvantagem o menor alcance e velocidade se comparado ao protocolo *WiFi*. Somados estes pontos foi escolhido o protocolo de comunicação *Bluetooth* para a criação da aplicação.

Em outras plataformas, como *iOS* e *Windows Phone 8*, o uso de comunicação *Bluetooth* sofre restrições e é limitado a uso de alguns periféricos, como fones de ouvido. Essas restrições impediriam o desenvolvimento do projeto proposto.

Para a compilação de aplicações para plataforma *Android*, fez-se os uso da *Android SDK* (*Software Development Kit*). Para o desenvolvimento de aplicações, pode-se fazer o uso do *JAVA*, linguagem nativa para aplicações para plataforma *Android*, ou linguagens *web*, com HTML, *Javascript* e CSS, suportadas por *frameworks* como *PhoneGap* e *Fries*, utilizados no desenvolvimento deste projeto.

O digrama do projeto é mostrado na *Figura 5*.

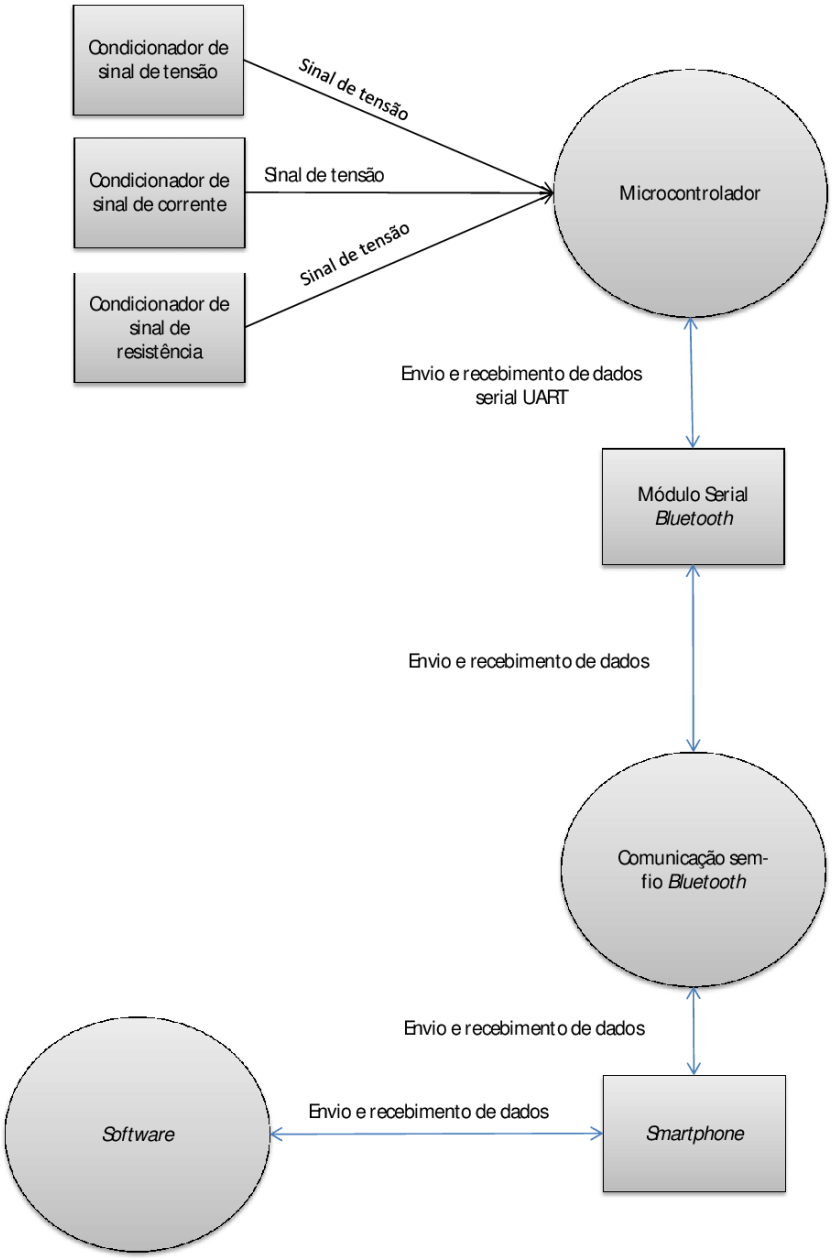


Figura 5 - Fluxograma do projeto

3.2 Materiais Utilizados

Para o desenvolvimento do projeto foi utilizado os componentes listados na *Tabela 1*. Já no *Apêndice F*, é apresentada uma tabela com o preço dos itens, mostrando o custo total.

Tabela 1 - Material Utilizado e Preços

Quantidade	Componente
1	Microcontrolador ATmega328P
2	Resistor Axial 10 kΩ 1% tolerância
2	Resistor Axial 1 kΩ 1% tolerância
1	Resistor Axial 1,2 kΩ 1% tolerância
1	Resistor Axial 5,6 kΩ 1% tolerância
1	Resistor Axial 1 Ω 0,5W 1% tolerância
1	Resistor Axial 56 Ω 1% tolerância
2	Capacitor cerâmico 22 pF 50V
1	Capacitor cerâmico 0,1 µF 50V
1	Capacitor cerâmico 0,33 µF 50V
1	Regulador de tensão LM7833
1	Cristal oscilador de 16 MHz
1	Amplificador Operacional LM358N
1	LED 5mm vermelho
1	Barra de pinos
1	Clip para bateria de 9 volts
1	Módulo *Bluetooth*
1	Placa de circuito impresso
1	Carcaça de acrílico
1	Par de pontas de prova
3	Borne vermelho para pino banana
1	Borne preto para pino banana
1	Chave 3 terminais 90 graus
1	Soquete para CI 8 pinos
1	Soquete para CI 28 pinos

Para a programação e compilação da aplicação foi utilizado um computador com sistema operacional *Linux*, com a distribuição *Linux Mint 16 MATE* versão 32-bit [11], e para testes, um *smartphone* com sistema *Google Android 4.4*.

A criação do *software* embarcado foi feita também na plataforma *Linux* utilizando a interface de desenvolvimento *Arduino IDE 1.0.5*. A programação do microcontrolador ATmega328P foi realizada com um programador, para circuitos de arquitetura AVR, *USBtinyISP* [12].

3.3 *Hardware*

Foi desenvolvida uma placa de aquisição e um *software* para dispositivos móveis, responsável pela visualização das medições elétricas realizadas.

O microcontrolador AVR ATmega328P utilizado trás como vantagens o baixo custo e simplicidade de uso e implementação, além de possuir periférico de comunicação serial UART (*Universal Asynchronous Receiver/Transmitter*), utilizado para interface com o módulo *Bluetooth*.

Um dos recursos existentes nesse microcontrolador é a presença de conversão de tensão analógica para valores de palavra digital segundo a tensão de referência, operando em uma faixa de tensão entre 1,8 V e 5,5 V, com uma resolução de 10 bits e 6 canais.

Devido a essa faixa de operação do circuito integrado ATmega328P, foi escolhida para alimentar o dispositivo a tensão de 3,3 V, própria tensão de operação do módulo *Bluetooth* empregado para comunicação com os dispositivos *Android*. Essa tensão é obtida utilizando o circuito integrado regulador de tensão LM7833, que fornece uma tensão fixa de saída de 3,3 V, a partir de uma tensão de entrada na faixa de valores entre 5,8 V e 18,3 V. Desta forma, a partir de um conjunto de quatro pilhas do tipo AA ou AAA de 1,5 V conectadas em série são suficientes para gerar um nível de tensão para operação do circuito. Uma bateria de 9 V também pode ser empregada para essa finalidade, além de fontes de tensão, que no entanto restringiriam o uso e fariam com que o dispositivo perdesse mobilidade. Com o intuito de comparação, caso fosse escolhido operar o circuito com níveis de tensão de 5 V, além da necessidade de

adequar os sinais lógicos para comunicação com o módulo *Bluetooth*, o uso de um regulador de tensão do tipo LM7855 faria necessária a uma associação de baterias que gerasse valores de tensão entre 7 V e 25 V, o que restringiria o uso de pilhas e baterias para a alimentação do circuito. Capacitores foram adicionados próximos ao regulador de tensão para melhorar a estabilidade de tensão e reduzir efeito de resposta transiente, e seus valores seguem as indicações presentes na folha de dados do circuito integrado.

A tensão de referência para o conversor analógico digital (V_{REF}) indica a faixa de conversão do periférico, atribuindo valores de palavra digital para os valores analógicos de entrada. Valores que excedam V_{REF} terão resultado códigos próximos ao código decimal 1023, ou 0x3FF em código hexadecimal.

V_{REF} pode ser selecionado entre três valores, AV_{CC} (tensão de alimentação dos circuitos analógicos), referência interna de 1,1V, e referência externa utilizando um pino externo AREF. AV_{CC} está conectada ao conversos analógico-digital (ADC), e a referência interna de 1,1V é gerada com os circuitos auxiliares que fornecem uma tensão precisa com o uso de um amplificador de instrumentação.

Conforme descrito, a tensão de operação do circuito como um todo é de 3,3 V, assim a tensão de referência utilizada foi a mesma empregada para alimentação do circuito. Foram empregados circuitos auxiliares para a adequação de valores de tensão cuja leitura se deseja fazer, bem como circuitos responsáveis pela transformação de corrente e resistência em valores de tensão, que serão lidos e convertidos pelo microcontrolador. A conversão do microcontrolador transforma valores de tensão em valores de palavra digital dada a tensão de referência. Sendo o conversor de 10 bits, a cada incremento de tensão no sinal de entrada no valor de $V_{ref}/2^{10}$ o valor digital acresce em uma unidade. A seguir, os cálculos para tensão de passo para um conversor AD de 10 bits com tensão de referência igual a 3,3 V:

$$V_{passo} = \frac{V_+}{2^{10}} = \frac{3{,}3}{1024} = 3{,}22266.10^{-3}\ V \quad (1)$$

Para a conexão de pontas de prova que serão utilizadas pelo usuário para efetuar as medições, foram utilizados bornes para pinos tipo “banana”, comumente empregados neste tipo de aplicação. A conexão com a bateria de 9 V, escolhida para a aplicação, é feita com um clip específico para esta função.

Os resistores empregados na montagem são axiais e dissipam ¼ de watt, com exceção do resistor shunt de 1 Ω que dissipa ½ watt, característica necessária para a função. Foi adicionado um LED (diodo emissor de luz) para indicar quando o circuito está ligado, e uma chave para acionamento do circuito.

Um cristal oscilador de 16MHz é utilizado pelo circuito como base de tempo e, para seu funcionamento, como indicado na folha de dados do microcontrolador ATmega328P, são associados dois capacitores cerâmicos de valor igual a 22 pF.

Já o módulo *Bluetooth* empregado, mostrado na *Figura 6* é baseado no circuito integrado BC417, fabricado pela CSR (*Cambridge Silicon Radio*), sendo este responsável pela comunicação.

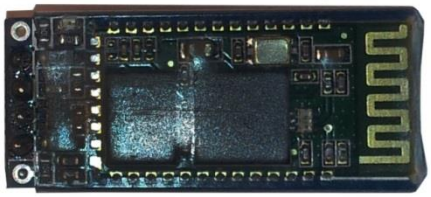


Figura 6 - Módulo *Bluetooth*

As características desse sistema são as seguintes:

- Senha padrão de pareamento: 1234;
- Tipo de comunicação: USART - Serial RS232 nível TTL;
- Frequência: 2,4 Ghz Banda ISM;
- Modulação: GFSK (*Gaussian Frequency Shift Keying*);
- Potencia de Transmissão: Classe 2, < 2,5 mW (4 dBm);
- Alcance: até 10 metros;
- Sensibilidade: < -84 dBm à 0.1% BER;
- Velocidade: assíncrona: 2,1 Mbps (Max) / 160 kbps, síncrona: 1 Mbps / 1 Mbps;
- Tensão: 3,6 - 6VDC 50 mA;
- Temperatura de trabalho: -20°C ~+75° C.

O circuito completo do dispositivo construído é mostrado no *Apêndice A*. A seguir, serão descritos separadamente os circuitos para a medição de tensão, corrente e resistência.

3.3.1 Voltímetro

Para a medição de tensão foi empregado um divisor resistivo, mostrado na *Figura 7*.

Figura 7 - Circuito para medição de tensão

Com este circuito, o limite de tensão de 3,3 V foi estendido para aproximadamente 30 V, conforme demonstrado nas equações:

$$i = \frac{V_{PProvaV}}{R_1 + R_2} \tag{2}$$

$$V_{sinalADv} = R_2 . i \tag{3}$$

$$V_{sinalADv} = \frac{R_2}{R_1 + R_2} . V_{PProvaV} \tag{4}$$

$$V_{sinalADv} = \frac{1{,}2k}{10k + 1{,}2k} . V_{PProvaV} = 107{,}143 . 10^{-3} . V_{PProvaV} \tag{5}$$

$$V_{PProvaV} = \frac{V_{sinalADv}}{107{,}143 . 10^{-3}} \tag{6}$$

$$V_{PProvaV\ máximo} = \frac{V_{sinalADv\ máximo}}{107{,}143 . 10^{-3}} \tag{7}$$

$$V_{sinalADv\ máximo} = 3{,}3\ V \tag{8}$$

$$V_{PProvaV\ máximo} = \frac{3{,}3}{107{,}143 . 10^{-3}} = 30{,}08\ V \tag{9}$$

$$V_{PProvaV\ passo} = \frac{30{,}8\ V}{1024} = \ 30{,}078\ mV \tag{10}$$

Considerando os resistores empregados na montagem do circuito, de 1% de tolerância, calcula-se o erro baseado neste valor. Dessa forma, se o resistor de 1,2 kΩ possuir 99% de seu valor nominal e o resistor de 10 kΩ, 101%, têm-se os seguintes valores:

$$V_{sinalADv} = \frac{1{,}2k.\,0{,}99}{10k.\,1{,}01 + 1{,}2k.\,0{,}99}.V_{PProvaV} = 105{,}245.\,10^{-3}.V_{PProvaV} \tag{11}$$

E portanto:

$$V_{PProvaV\ máximo} = \frac{3{,}3}{105{,}245.\,10^{-3}} = 31{,}356\ V \tag{12}$$

$$V_{PProvaV\ passo} = \frac{31{,}3556\ V}{1024} = 30{,}621\ mV \tag{13}$$

$$Erro_{relativo\ máximo} = \frac{31{,}3556V - 30{,}08V}{31{,}3556V} = 1{,}77\% \tag{14}$$

3.3.2 Amperímetro

Para a medição de valores de corrente, mostrado na *Figura 8*, foi utilizada uma topologia com o uso de amplificador operacional que faz a leitura de valores de tensão sobre um resistor *shunt* de 1 Ω. Está montagem foi criada seguindo princípios para medição de corrente, conforme nota de aplicação fornecida pela *Linear Technology* [13]. Assim, o valor de corrente que atravessa este resistor gera uma queda de tensão proporcional ao valor do resistor *shunt*.

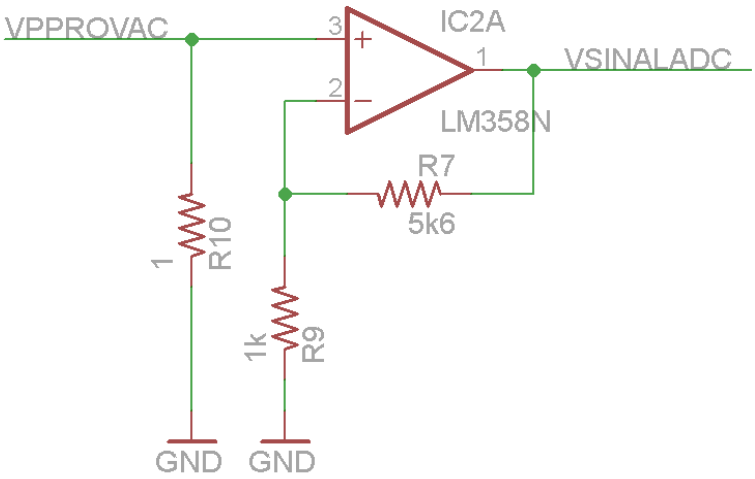


Figura 8 - Circuito para a medição de corrente

Foi empregado um resistor *shunt* de 1 Ω de forma a introduzir baixa impedância ao circuito cuja leitura será realizada, e a topologia com o uso de um amplificador operacional foi o de um amplificador não inversor, com um ganho que adequa o valor de tensão sobre o resistor shunt para a entrada do circuito conversor analógico/digital presente no microcontrolador.

As principais especificações do amplificador operacional usado, o LM358P, fabricado pela *Texas Instruments*, são:

- Alimentação simples: 3 V – 32 V;
- Alimentação dupla: ±1,5 V – ±16 V;
- Baixa tensão de offset de entrada: 2 mV;
- Elevado ganho de tensão DC: 100 dB;
- Largura de banda para ganho unitário: 1 MHz;
- Temperatura de operação: 0º C a 70º C;
- Dissipação de potência: 830 mW.

Abaixo estão apresentadas as contas que descrevem o circuito projetado:

$$V_{sinalADc} = V_{PProvaC}.\left(1 + \frac{R_7}{R_9}\right) = V_{PProvaC}.\left(1 + \frac{5{,}6k}{1k}\right) = V_{PProvaC}.6{,}6 \quad (15)$$

$$V_{sinalADc\ máximo} = 3{,}3\ V \quad (16)$$

$$V_{PProvaC\ máximo} = \frac{V_{sinalADc\ máximo}}{6{,}6} = 500\ mV \quad (17)$$

$$i_{PProva\ max} = \frac{500.10^{-3}}{R_S} = \frac{500.10^{-3}}{1} = 500\ mA \quad (18)$$

$$i_{PProva\ passo} = \frac{500\ mA}{1024} = 488{,}281\ \mu A \quad (19)$$

Considerando os resistores empregados na montagem do circuito, de 1% de tolerância, calcula-se o erro baseado neste valor. Dessa forma, considerando que o resistor de 5,6 kΩ possua 99% de seu valor nominal e o resistor de 1 kΩ 101% têm-se os seguintes valores:

$$V_{sinalADc} = V_{PProvaC}.(1 + \frac{5{,}6k.0{,}99}{1k.1{,}01}) = V_{PProvaC}.6{,}48911 \quad (20)$$

E portanto:

$$V_{sinalADc\ máximo} = 3{,}3\ V \quad (21)$$

$$V_{PProvaC\ máximo} = \frac{V_{sinalADc\ máximo}}{6{,}48911} = 508{,}544\ mV \quad (22)$$

$$Erro_{relativo\ máximo} = \frac{508{,}544mV - 500mV.0{,}99}{508{,}544mV} = 2{,}663\% \quad (23)$$

3.3.3 Ohmímetro

Para a leitura de resistência, também foi utilizado um divisor resistivo, *Figura 9*.

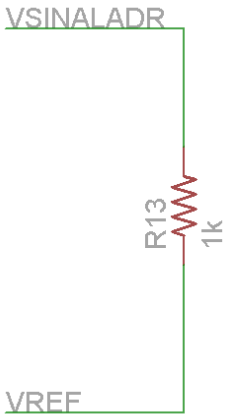


Figura 9 - Circuito para a medição de resistência

Diferentemente da medição de tensão, aqui o valor de tensão é fixo a referência de 3,3 V e, assim, estima-se o valor de resistência pela queda de tensão que este gera em associação com um resistor de valor fixo em 1 kΩ.

$$V_{sinalADr} = \frac{R_M}{R_1 + R_M}.V_{ref} \tag{24}$$

$$V_{ref} = 3{,}3V \tag{25}$$

$$V_{sinalADr} = \frac{R_M}{R_1 + R_M}.3{,}3 \tag{26}$$

$$V_{sinalADr}.(R_1 + R_M) = R_M.3{,}3 \tag{27}$$

$$R_M = \frac{V_{sinalADr}.R_1}{3{,}3 - V_{sinalADr}} \tag{28}$$

$$V_{sinalADr\ mínimo} = 0\ V \tag{29}$$

$$R_{M\ mínimo} = \frac{0.1k}{3{,}3 - 0} = 0\ \Omega \tag{30}$$

$$V_{sinalADr\ máximo} = 3{,}3 - (\frac{3{,}3}{1024}) = 3{,}297\ V \tag{31}$$

$$R_{M\ máximo} = \frac{3{,}29678.1k}{3{,}3 - 3{,}29678} = 1023000\ \Omega \tag{32}$$

$$R_{M\ passo} = \frac{3{,}22266.10^{-3}.1k}{3{,}3 - 3{,}22266.10^{-3}} = 977{,}517\ m\Omega \tag{33}$$

Considerando os resistores empregados na montagem do circuito, de 1% de tolerância, calcula-se o erro baseado neste valor. Assim, considera-se que o resistor de referência de 1 kΩ possua 101% de seu valor nominal, obtêm-se os seguintes valores:

$$R_{M\ máximo} = \frac{3{,}29678.1k.1{,}01}{3{,}3 - 3{,}29678} = 1034083{,}168\ \Omega \tag{34}$$

E portanto:

$$Erro_{relativo\ máximo} = \frac{1034083{,}1677 - 1023000}{1034083{,}1677} = 1{,}072\% \tag{35}$$

3.3.4 Construção do Projeto

Utilizando o *software CADSoft Eagle PCB Design* [14], foi desenhado o *layout* para ser transferido para uma placa de circuito impresso, que foi posteriormente corroída, perfurada e limpa para a soldagem dos componentes. O desenho da placa foi feito manualmente por meio das ferramentas disponibilizadas pelo programa. O *layout* final é mostrado na *Figura 10*.

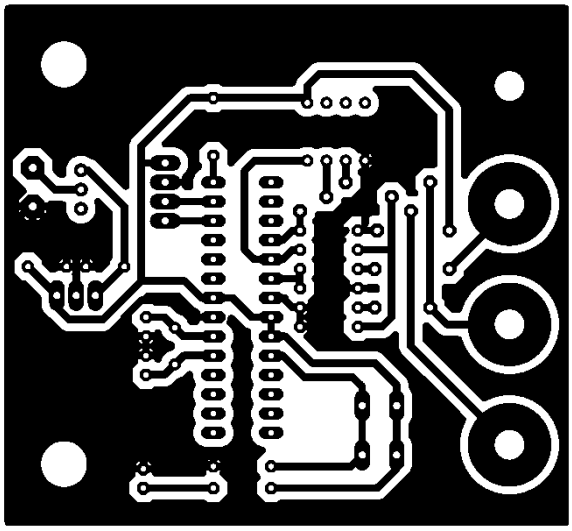


Figura 10 - Layout da placa de circuito impresso

A máscara de componentes da placa pode ser vista na *Figura 11*.

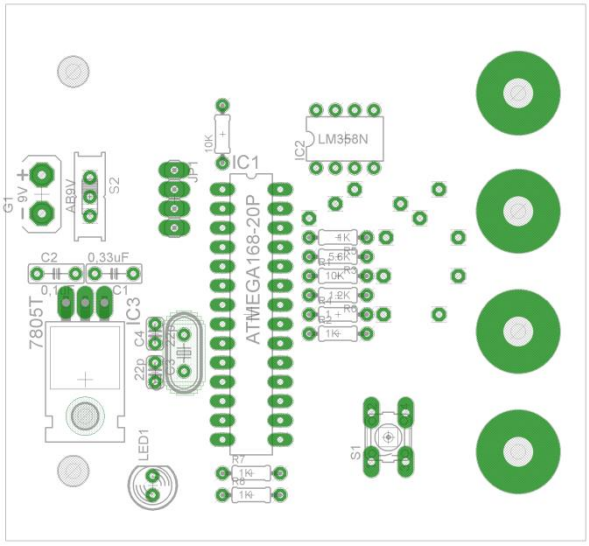


Figura 11 - Máscara de componentes da placa de circuito impresso

Antes da confecção da placa de circuito impresso, um modelo tridimensional foi desenhado no *software Google SketchUp* [15], com a finalidade de prever possíveis problemas de *design* de *layout*. Este modelo é mostrado na *Figura 12* e *Figura 13*.

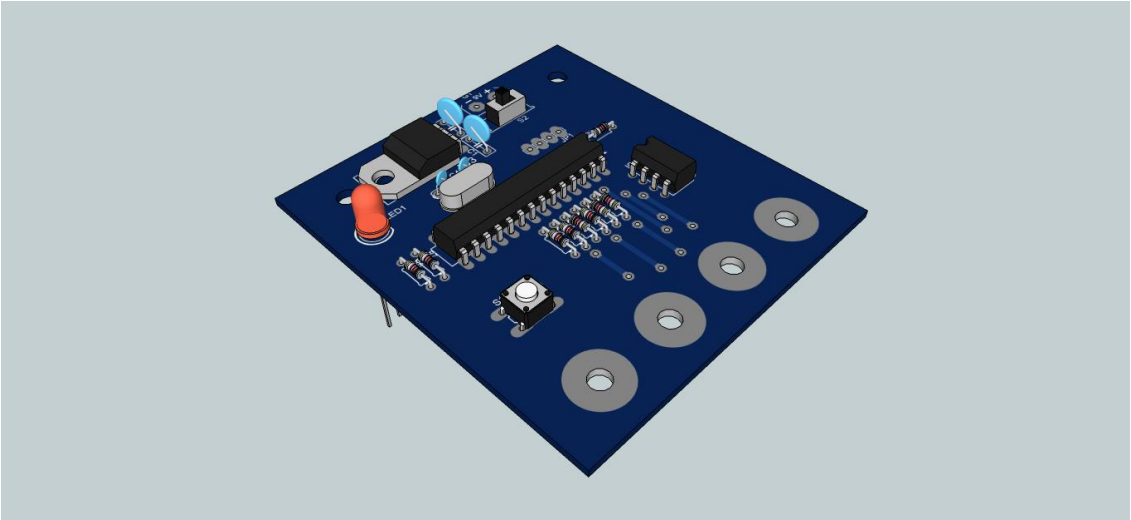


Figura 12 - Vista de cima do modelo tridimensional da placa

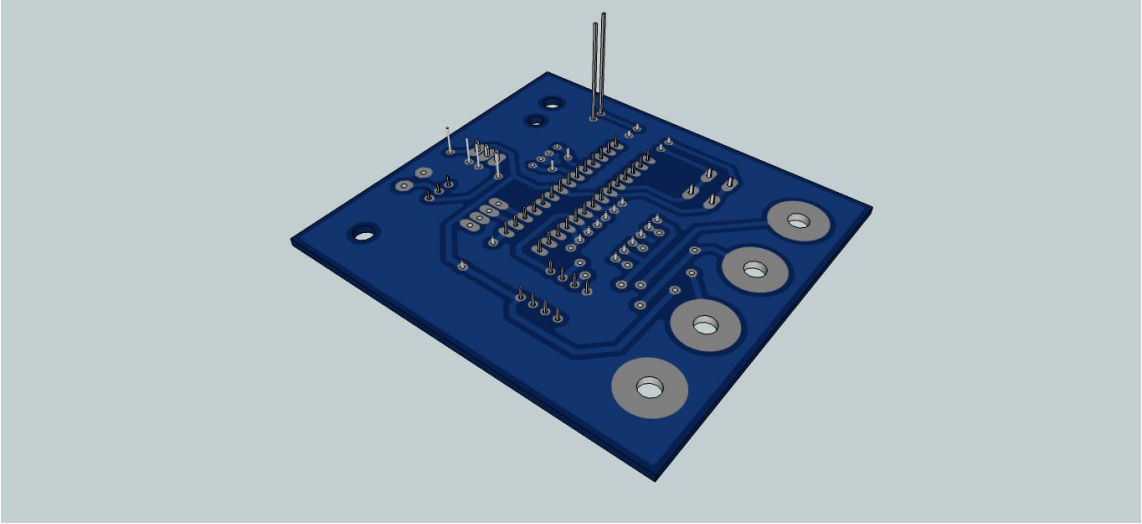


Figura 13 - Vista de baixo do modelo tridimensional da placa

3.3.5 Montagem da Placa

Foi realizada a confecção da placa de circuito impresso. A *Figura 14* mostra a placa após a corrosão já limpa e perfurada, restando o processo de soldagem dos componentes.

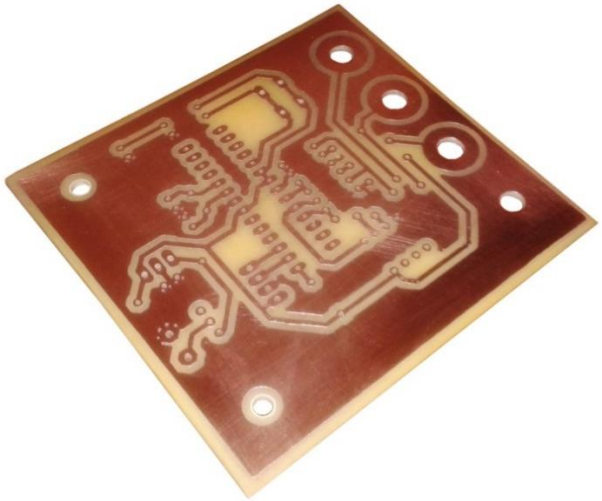


Figura 14 - Placa de circuito impresso

A *Figura 15* mostra a parte superior da placa já com todos os componentes soldados. Uma visão da parte inferior da placa pode ser vista na *Figura 16*.

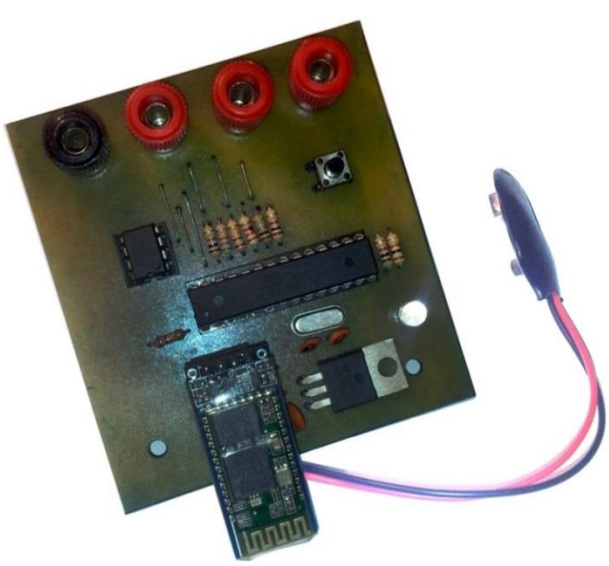


Figura 15 - Vista superior do circuito completo em placa de circuito impresso

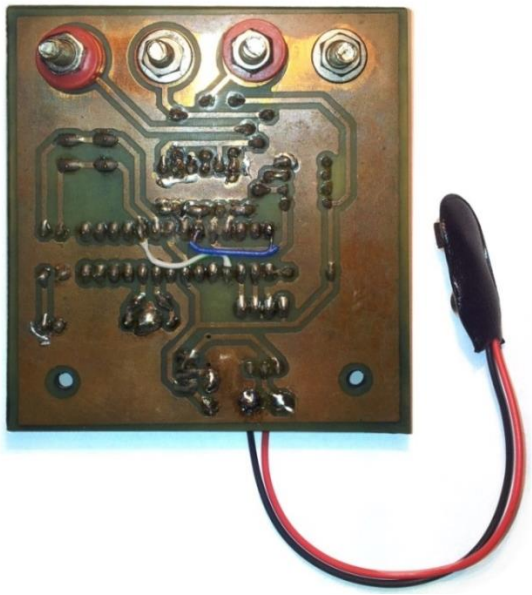


Figura 16 - Vista inferior do circuito completo em placa de circuito impresso

Tendo comprovado sua funcionalidade, o circuito proposto foi então revisto e modificado de forma atender de melhor forma a aplicação, novamente foi feito o uso do *software CADSoft Eagle PCB Design*. A seguir são listadas as modificações realizadas:

- Retirada da chave táctil do circuito que era empregada na primeira versão do *software* para plataforma *Android* para escolha da medida a ser realizada, opção que agora é feita exclusivamente utilizando o *software*.
- Mudança de *layout* da placa que permitiu sua diminuição, com o uso de placa de dupla camada e nova disposição de componentes, além da adequação para uso de uma caixa para proteção.
- Adição de um conector para programação ICSP (*In-circuit serial programming*) para facilitar a gravação e atualização do programa desenvolvido para o microcontrolador.
- Uso de resistores de 1% de tolerância, para a realização de medições mais precisas.

O *layout* final da nova placa é mostrado na *Figura 17*. Uma visão da parte superior e inferior com os componentes já soldados pode ser visto nas *Figuras 18* e *19*.

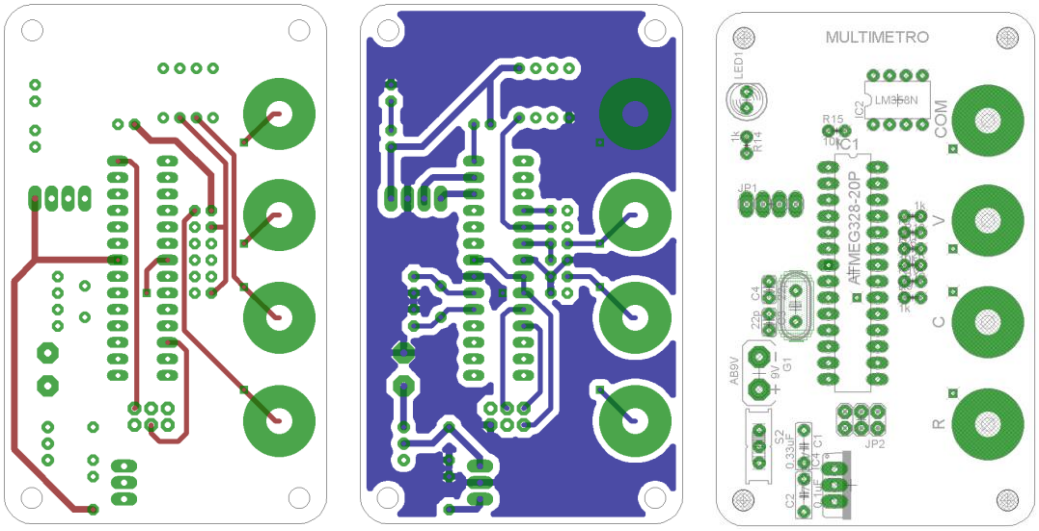


Figura 17 - Circuito Revisado

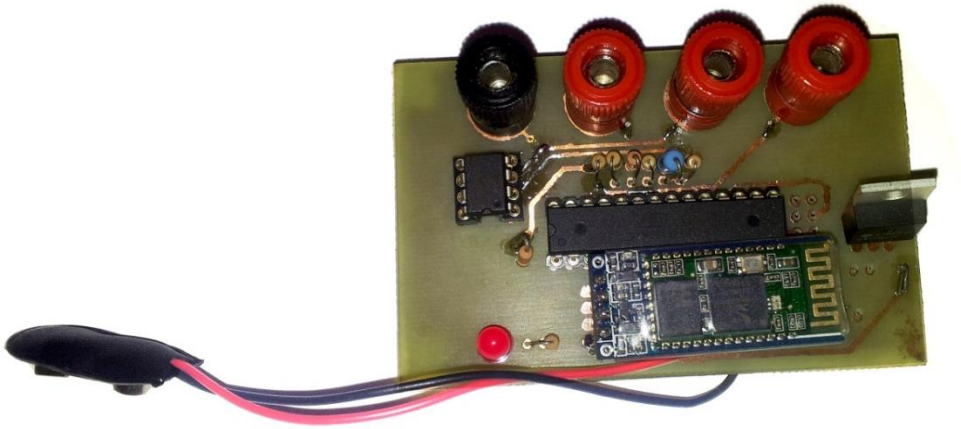


Figura 18 - Vista superior da placa revisada

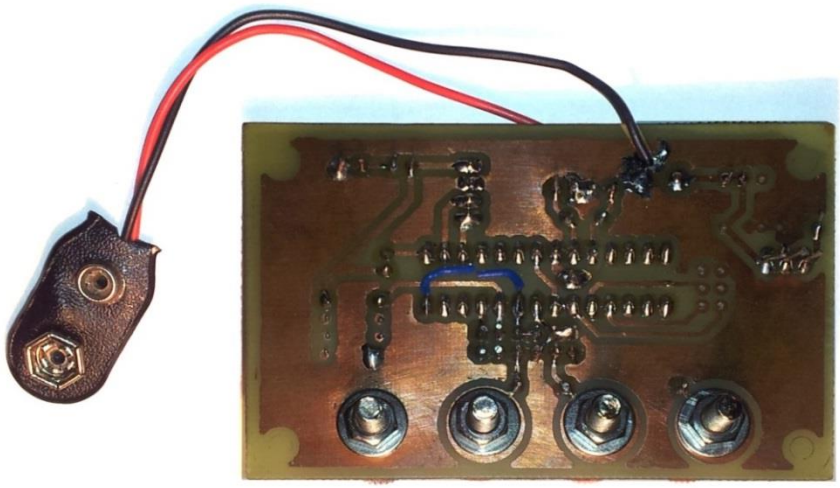


Figura 19 - Vista inferior da placa revisada

3.4 *Software*

3.4.1 Preparação do ambiente de trabalho para o *software*

Para a criação da aplicação proposta foi decidido o uso de um sistema operacional baseado em *Linux*, já que a maioria dos recursos empregados para o desenvolvimento da aplicação é de código aberto e possui maior suporte e documentação neste sistema. A distribuição escolhida foi a *Linux Mint*, baseada em *Debian* e *Ubuntu*.

Para o desenvolvimento e compilação de aplicações para plataforma *Google Android* é necessária a instalação da *Android SDK* (*Software Development Kit*), ferramenta que fornece as bibliotecas e componentes necessários para a compilação, teste e depuração de aplicações desenvolvidas para o sistema *Android*. Esta ferramenta requer a instalação do *Oracle Java SDK*. Após obter os arquivos necessários, a instalação é feita com os comandos mostrados na *Figura 20* via terminal na pasta onde os instaladores estiverem presentes:

```
#  chmod a+x jdk-6u37-linux-i586.bin
#  ./jdk-6u37-linux-i586.bin
#  tar -xvf jdk-7u7-linux-i586.tar.gz
#  unzip jdk-6u30-apidocs.zip -d jdk1.6.0_37/
#  unzip jdk-7u6-apidocs.zip -d jdk1.7.0_07/
#  tar -xvf jdk-6u37-linux-i586-demos.tar.gz
#  tar -xvf jdk-7u9-linux-i586-demos.tar.gz
#  sudo mv jdk1.6.0_37 /usr/lib/jvm
#  sudo mv jdk1.7.0_07 /usr/lib/jvm
#  chmod +x update-java-0.5b
#  sudo ./update-java-0.5b
```

Figura 20 – Lista de comandos para a instalação das ferramentas utilizadas

Obtido o arquivo de instalação do *Android SDK*, a instalação é feita com o uso dos comandos mostrados na *Figura 21* via terminal *Linux* na pasta em que os arquivos estiverem presentes.

```
# mv android-sdk-linux /usr/lib/
# Cd /usr/lib/android-sdk-linux-/tools
# ./android
```

Figura 21 – Comandos para a instalação do *Android SDK*

Esse comando executará o gerenciador de pacotes do *Android SDK*, *Figura 22*, onde é feita a seleção para a instalação e atualização dos pacotes para a versão alvo do *Android*.

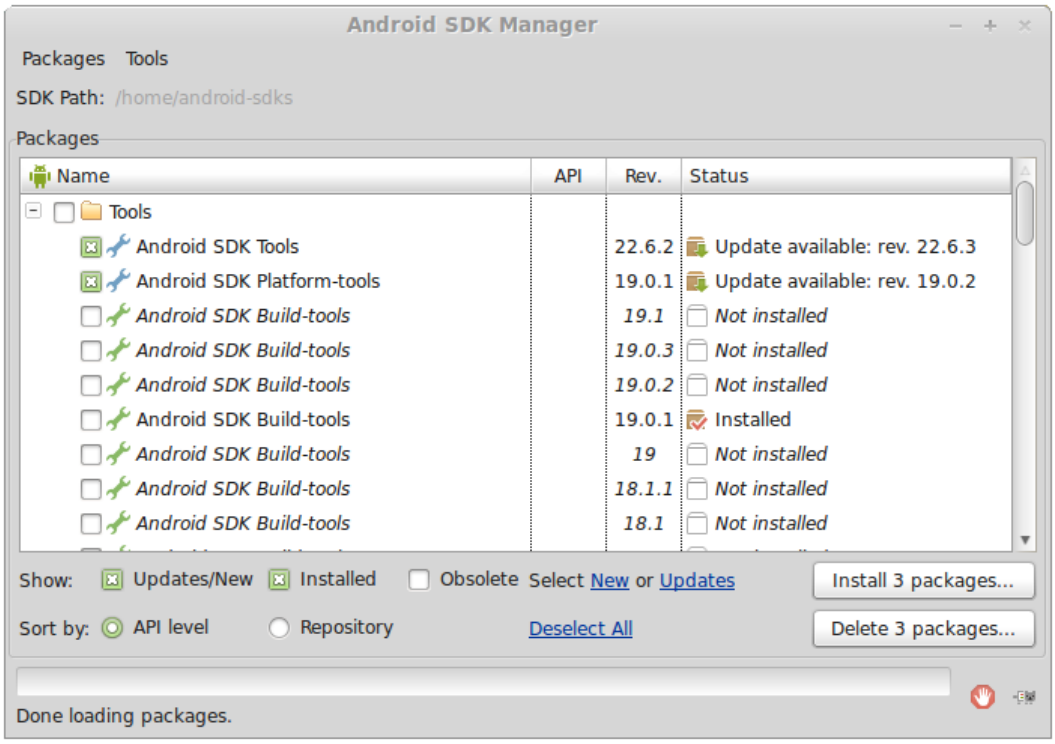


Figura 22 - Tela do *Android SDK Manager*

Para a instalação das ferramentas que serão utilizadas no desenvolvimento da aplicação é utilizado o gerenciador de pacotes NPM [16], presente no pacote de instalação do *Node.js* [17], cujo arquivo de instalação pode ser obtido na página do *Node.js* na internet. A ferramenta *PhoneGap* tem como dependência a plataforma *Apache-cordova* que deve ser instalada em conjunto.

Abaixo, na *Figura 23*, os passos e comandos utilizados em terminal no *Linux* utilizando o gerenciador NPM:

```
# npm install -g cordova
# npm install -g phonegap
```

Figura 23 – Comandos para instalação do *framework PhoneGap*

Já com as ferramentas necessárias instaladas, a criação da aplicação para *Android* com o *PhoneGap*, além da instalação dos *plugins* empregados e compilação, é feita por meio dos comandos, conforme *Figura 24*, no terminal:

```
# phonegap create multimetro
# cd multímetro
# cordova platform add android
# cordova plugin add https://github.com/don/
BluetoothSerial
# cordova plugin add
https://github.com/EddyVerbruggen/SocialSharing-PhoneGap-
Plugin
# cordova build android
```

Figura 24 – Comandos para a criação de uma aplicação no *PhoneGap*

Após execução dos comandos listados, os arquivos necessários para a criação da aplicação estão presentes na pasta “multímetro”, nome escolhido para o projeto. Neste diretório, encontram-se, dentre várias, as seguintes pastas:

- platforms: diretório que contém os arquivos de configuração da plataforma para a qual será feita a aplicação, neste caso, *Android*, além de armazenar sua compilação.
- plugins: diretório que armazena os *plugins* instalados e que são utilizados no projeto.
- www: a qual contém os arquivos em linguagem *web* na qual será feita a programação.

Dentro da pasta “www” está presente o arquivo “config.xml”. Nneste arquivo são especificados detalhes, como o nome da aplicação, a sua descrição e preferências de execução, como por exemplo a execução em tela cheia no dispositivo.

Foi criado um diretório dentro da pasta “www”, chamado “lib”, no qual foram adicionados arquivos utilizados na criação da aplicação, como ícones, além dos arquivos de programação em CSS e *JavaScript*. Para cada tipo de arquivo foi criada uma pasta visando maior organização do projeto. Os arquivos da interface de usuário *Fries*, baixados em seu repositório na internet, foram adicionados nesta pasta, para serem usados na aplicação. Para isso, é necessária a referência aos arquivos HTML criados para a aplicação que utilizarão os componentes desta ferramenta, como exemplificado na *Figura 25*.

```
<link rel="stylesheet" href="lib/css/base.css">
<link rel="stylesheet" href="lib/css/action-bars.css">
<link rel="stylesheet" href="lib/css/chevrons.css">
<link rel="stylesheet" href="lib/css/tabs.css">
<link rel="stylesheet" href="lib/css/content.css">
<link rel="stylesheet" href="lib/css/buttons.css">
<link rel="stylesheet" href="lib/css/forms.css">
<link rel="stylesheet" href="lib/css/lists.css">
<link rel="stylesheet" href="lib/css/spinners.css">
<link rel="stylesheet" href="lib/css/icomoon.css">
<link rel="stylesheet" href="lib/css/stack.css">
<link rel="stylesheet" href="lib/css/sliders.css">
```

Figura 25 – Referências aos componentes que serão utilizados no arquivo HTML

Para o desenvolvimento do software embarcado foi escolhida a plataforma *Arduino*, que permite a rápida prototipagem de circuitos microcontroladores e possui uma interface simples e intuitiva. A instalação da plataforma pode ser feita diretamente em distribuições *Linux* baseadas em *Debian* utilizando o comando da *Figura 26*.

```
#  apt-get install arduino
```

Figura 26 – Comando para instalação da *IDE Arduino*

Para a gravação do microcontrolador foi utilizado um programador AVR e interface SPI chamado *USBtinyISP*, plataforma de arquitetura aberta USB suportado nativamente pela *Arduino IDE* (*Integrated Development Environment*), *Figura 27*.

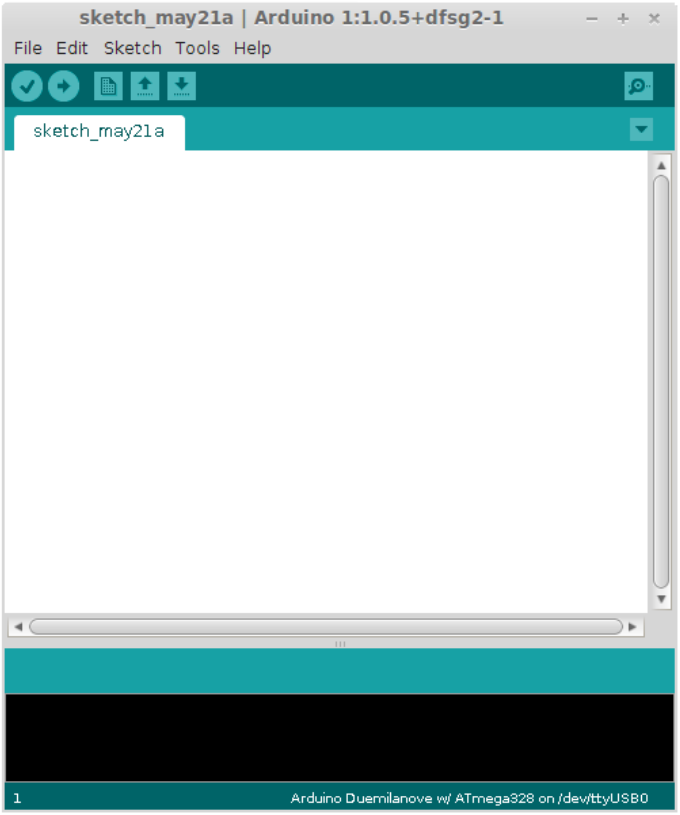


Figura 27 – Tela Arduino IDE

3.4.2 Programa para o microcontrolador

O programa embarcado, desenvolvido com o auxílio da plataforma de código aberto *Arduino*, consiste em um sistema para seleção da medida a ser realizada, podendo ela ser tensão, corrente ou resistência elétrica, que é feita por meios de 3 canais de conversão analógico/digital presentes no microcontrolador Atmega328P.

A seleção da medida de interesse é feita pelo envio do dispositivo *Android* de um carácter de seleção, 'T' para tensão, 'C' para corrente, e 'R' para resistência, utilizando o módulo *Bluetooth*, que opera conforme protocolo serial. Desta forma, utilizando as funções de comunicação serial já presentes na plataforma *Arduino*, o envio

e recebimento de dados é feito facilmente. Assim, conforme requisitado pelo aplicativo do dispositivo móvel, a placa realiza as medições de interesse e as envia de forma contínua.

A fim de se realizar medições mais precisas, foi empregado um algoritmo que utiliza a referência interna de 1,1 V, *Figura 28*, para garantir uma leitura mais próxima do valor real. Essa rotina é executada toda vez que se deseja fazer uma leitura, tanto para tensão, corrente, ou resistência e calcula o valor máximo, que corresponde a palavra digital igual a 1023, baseado na referência de 1,1 V, seguindo um algoritmo bastante simples, mas eficiente. Utilizando o valor de referência de 1,1 V, lido primariamente, é então medido o valor de AREF e comparado a este, assim, para o cálculo da conversão o valor de referência externa é atualizado antes de cada medição, assim as flutuações que podem influir sobre o circuito são minimizadas.

```
long Vref() {
  //Usa um circuito de referência interna do
microcontrolador para fazer medições mais precisas
  long resultado;
  // Lê valor de 1.1V de referência contra AVcc
  ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  delay(2); // Espera Vref estabilizar
  ADCSRA |= _BV(ADSC); // Conversão
  while (bit_is_set(ADCSRA,ADSC));
  resultado = ADCL;
  resultado |= ADCH<<8;
  resultado = 1125300L / resultado; // Recalcula AVcc em
escala de mV
  return resultado;  // Retorna o valor de referência
}
```

Figura 28 – Algoritmo para a realização de medições mais precisas

Para a comunicação com os dispositivos *Android*, a interface com o módulo *Bluetooth* empregado é feito via comunicação serial UART (*Universal Asynchronous Receiver/Transmitter*). Assim, foram utilizadas as funções para o uso deste periférico na plataforma *Arduino*. Inicialmente a comunicação é inicializada a partir da taxa de transferência com a qual se deseja operar. Neste caso, foi escolhida a taxa de 38400 bits por segundo que é a mesma taxa utilizada pelo módulo *Bluetooth* empregado. Uma função recursiva é responsável pela leitura do canal de conversão analógico digital da

medida de interesse, armazenada via recebimento da escolha do usuário, e envio para aplicação *Android*.

O diagrama da *Figura 29* mostra o fluxo de execução do programa desenvolvido para o microcontrolador. Este programa completo pode ser visto no *Apêndice B.*

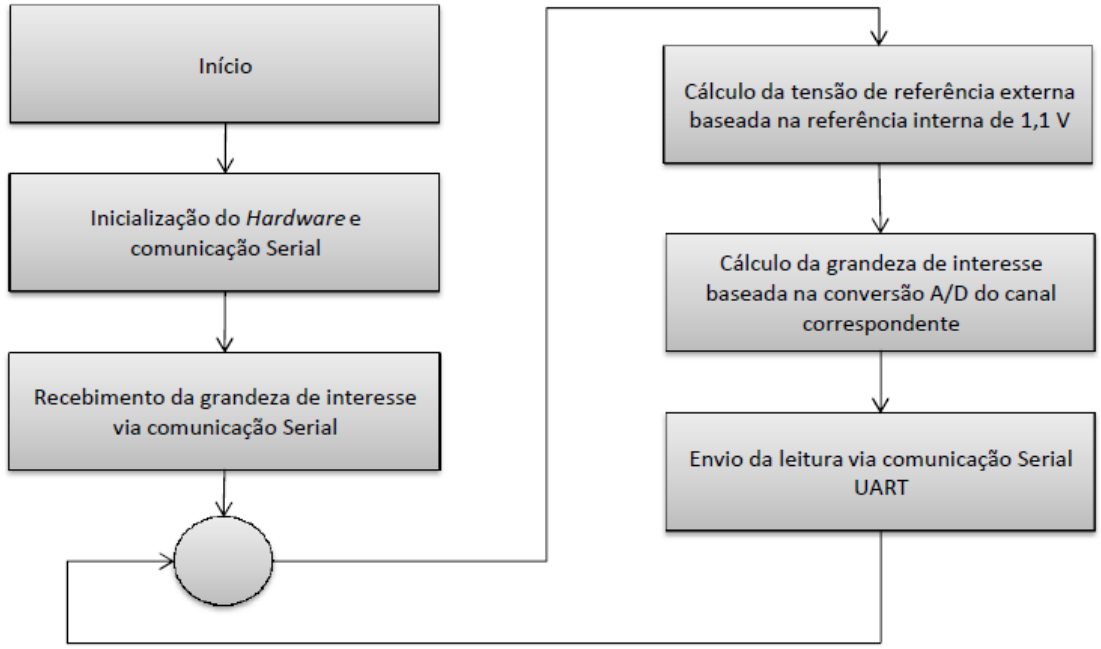


Figura 29 - Fluxograma do programa do microcontrolador

3.4.3 Programa para *Android*

Como proposta inicial, o *software* deveria receber de um dispositivo, por meio de conexão sem fio utilizando o protocolo *Bluetooth*, uma série de medidas elétricas (tensão, corrente e resistência) e exibi-los na tela do dispositivo *Google Android* em que estivesse sendo executado, além de exibir valor máximo e mínimo da medida realizada em um intervalo de tempo.

Foi escolhida então a ferramenta *PhoneGap* para a criação do aplicativo, por permitir o acesso a grande quantidade de recursos de *hardware* dos dispositivos *Android*, além de suportar a expansão por meio de *plugins*.

O primeiro passo a ser tomado foi a implementação da comunicação *Bluetooth*, o que pode ser feito com a instalação de um *plugin* que permite a comunicação serial, sendo, portanto, compatível com o módulo escolhido.

Para a criação de uma interface de usuário foi escolhida a linguagem HTML por ser suportada pelo *PhoneGap* e por trazer como principal vantagem uma escrita simples e fácil implementação. Para facilitar a criação da interface de usuário, foi utilizado o *framework Fries* que possui identidade visual inspirada no contexto padrão do *Android 4.X*. Tal ferramenta possui componentes como botões, barras de ação, navegação por *tabs* além de outros recursos facilmente customizáveis por meio dos arquivos HTML e CSS.

Terminada esta primeira versão, que somente exibia os valores medidos, e controlava o tipo de medição de interesse, foi proposta a exibição das medidas de tensão e corrente, por meio de um gráfico. Desta forma, leituras realizadas durante uma janela de tempo seriam armazenadas e então a curva de medições plotada em um gráfico de dois eixos. Como solução para a criação deste gráfico, foi utilizada a biblioteca *Flot* [18], que permite a construção de gráficos em aplicações para linguagem *web*, como HTML. Esta biblioteca têm como dependência a ferramenta *jQuery* [19], uma biblioteca *JavaScript* de arquitetura aberta que torna simplificado o processo de desenvolvimento de aplicações *web* executadas pelo cliente da aplicação. Uma versão desta ferramenta para plataformas móveis é chamada *jQuery Mobile* [20]. Esta biblioteca possui recursos muito úteis que foram utilizados na aplicação desenvolvida, como ajuste de eixo, que permite uma maior resolução de exibição das leituras, baseado no valor máximo durante a janela amostrada.

Outro recurso utilizado foi o da atualização em tempo real do gráfico, dando a sensação de uma amostragem contínua, a cada novo recebimento de medições.

Por fim, foi proposto o armazenamento das leituras realizadas de forma a permitir que o usuário as utilize de forma posterior, para análise ou registro. Este recurso foi possível por meio do suporte a arquivos nativo da ferramenta *PhoneGap*. O formato de armazenamento segue o padrão de criação de variáveis do tipo vetor, suportado por diversas linguagens de programação, como, por exemplo, o *Matlab*. Portanto, se o usuário tiver interesse, pode manipular estes dados em uma série de aplicações de forma simples.

Utilizando novamente o suporte do *PhoneGap* a *plugins*, foi implementado uma extensão que trás um recurso interessante por meio da interface da *widget* de

compartilhamento presente nas versões do *Android 4.X*. Com o *plugin SocialSharing*, o usuário pode compartilhar as medidas realizadas por meio de armazenamento online, utilizando os serviços *Google Drive* ou *Dropbo*x por exemplo, ou então como anexo de uma mensagem de e-mail, tendo, desta forma, uma variedade de serviços para armazenar as medições feitas e acessar quando desejado.

Ao iniciar o aplicativo, a primeira tela com a qual o usuário tem contato é responsável pela interface para conexão com o dispositivo *Bluetooth*, como mostra a *Figura 30* a seguir.

Figura 30 - Tela de conexão *Bluetooth*

Caso a conexão *Bluetooth* do aparelho estiver desativada, o usuário terá então que ativá-la nas configurações de rede sem fio. Além disso, o pareamento com o dispositivo deve ser feito de forma prévia pelo usuário, de forma a adicionar a placa à lista de opções de conexão. Ele é feito acessando o menu de configurações no sistema *Android*, na seção de redes sem fio. Após habilitação do protocolo sem fio *Bluetooth*, é feita a procura de dispositivos ligados próximos ao aparelho, e o pareamento ocorre após escolha e inclusão de senha caso necessário. O módulo empregado possui "BT UART" como nome padrão e senha "1234". Confirmado o pareamento, o usuário pode iniciar a aplicação de multímetro e selecionar a placa clicando na opção "BT UART", agora presente. Este processo é mostrado na *Figura 31* a seguir.

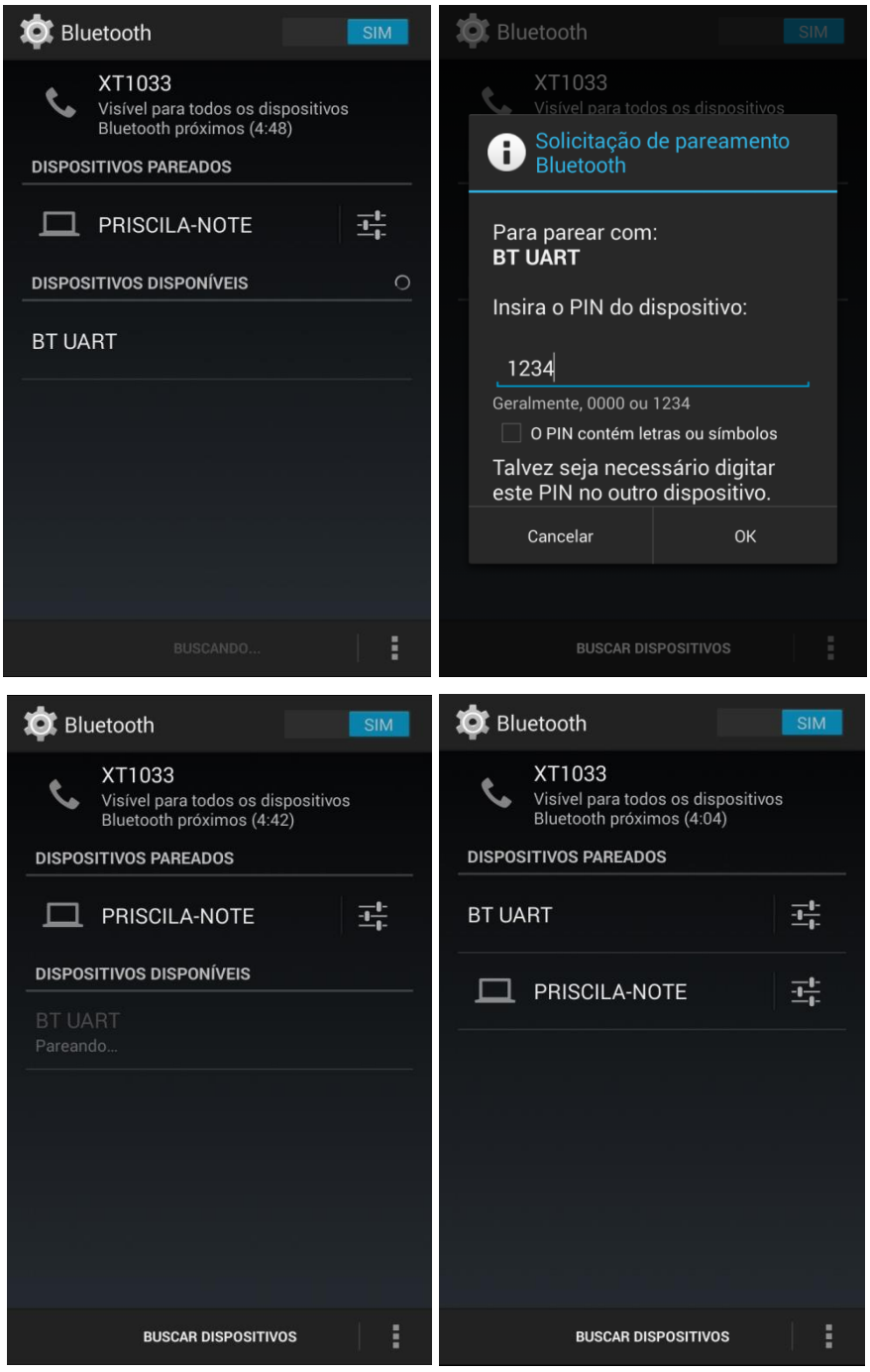


Figura 31 - Telas de pareamento *Bluetooth*

Após ativada a conectividade via *Bluetooth*, o usuário se depara com uma lista em que seleciona o dispositivo com o qual deseja se conectar, como na *Figura 32*. Deve-se atentar ao fato de que somente são exibidos dispositivos que foram previamente pareados pelo usuário.

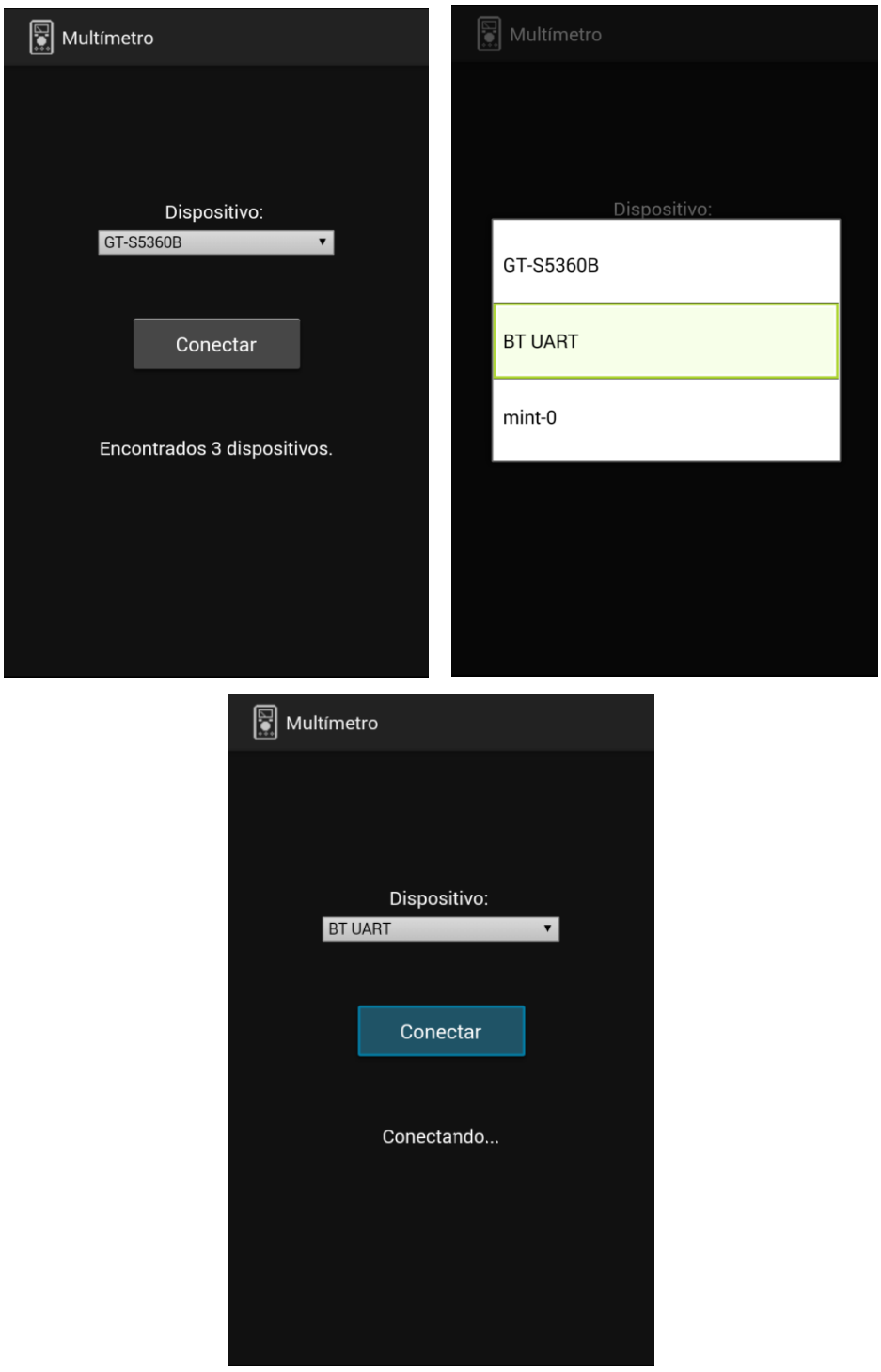


Figura 32 - Telas de conexão com o dispositivo desejado

Ao pressionar o botão "Conectar" e sendo efetiva esta conexão, o aplicativo executa uma segunda tela, como pode ser visto na *Figura 33*. Nesta tela são exibidos os valores de medição instantânea, além dos valores máximo e mínimo aparentes em uma janela de tempo, a mesma que é exibida no gráfico. Também estão presentes dois botões

para seleção da *tab* ativa. Entre as opções têm-se "Leitura" e "Gráfico", sendo a primeira ativa por padrão.

Figura 33 - Tela de Leitura

Na parte inferior, estão presentes três botões para a escolha da medição a ser realizada. Dentre as opções disponíveis estão a de tensão, corrente ou resistência, o que configura na seleção do borne a ser lido. Foi definido que sempre que a aplicação for iniciada a medição ativa por padrão é a da tensão.

Portanto para a medição de outra grandeza, o usuário deverá trocar as pontas de prova para os bornes de interesse e pressionar na tela do aplicativo a escolha realizada.

Caso o usuário deseje ver em um gráfico o comportamento da medida realizada, basta pressionar na parte superior o botão "Gráfico", que faz a troca da *tab* ativa. A tela pode ser vista na *Figura 34* a seguir.

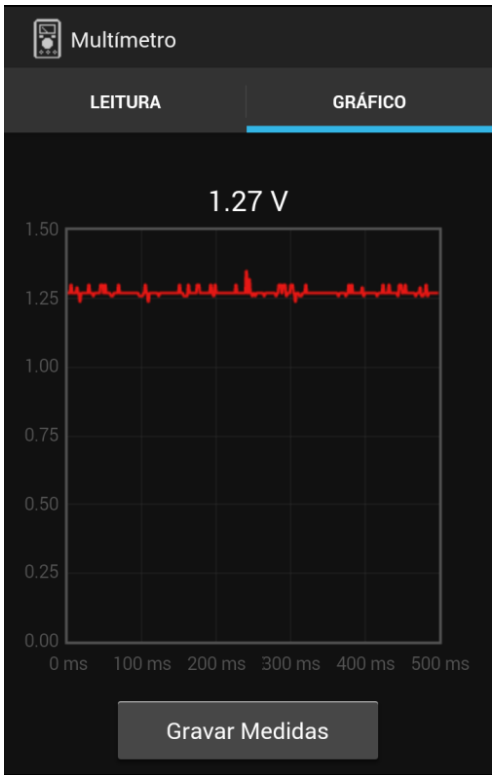


Figura 34 - Tela do Gráfico

Na parte superior da *tab* "Gráfico", a medição atual é exibida de forma a orientar o usuário da grandeza que está sendo lida e plotada.

Foi definida a não exibição do gráfico caso a medição realizada seja a de resistência, uma vez que a variação desta grandeza ao longo do tempo não representa uma medida de interesse, já que na maioria dos casos se mantém constante. Assim somente é plotado o gráfico para medições de tensão, ou corrente.

Também se faz presente na *tab* “Gráfico” um botão para gravação das medidas realizadas, mais especificamente as que estão sendo exibidas na janela correspondente ao gráfico. Ao pressionar o botão “Gravar Medidas” o usuário tem a opção de escolher dentre diversos serviços para o compartilhamento do arquivo gerado, como mostra a *Figura 35*.

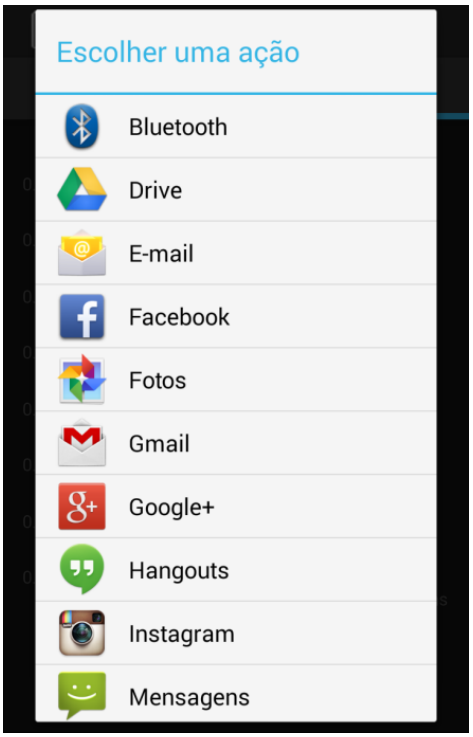


Figura 35 - Tela de compartilhamento do arquivo

O início do desenvolvimento da aplicação para sistema *Google Android* começou com a criação das telas de interface de usuário. Como dito anteriormente, foi empregado no desenvolvimento uma interface baseada no estilo visual predominante do sistema *Android 4.X*, chamada *Fries*.

Utilizando a documentação presente no *site* onde está disponível, foram utilizadas as funções em HTML para a criação conforme a necessidade da inclusão dos elementos de interesse.

Para o uso das funções e estilos disponibilizados pelo *Fries*, é necessário referenciar no arquivo HTML o caminho para as dependências utilizadas, como os arquivos CSS, e funções em *JavaScript*, *Figuras 36 e 37*. Entre esses arquivos, também estão presentes o arquivo CSS da aplicação, nomeada como *index.css*, e arquivo *Javascript*, *index.js*:

```
<link rel="stylesheet" href="lib/css/index.css">
<link rel="stylesheet" href="lib/css/base.css">
<link rel="stylesheet" href="lib/css/action-bars.css">
<link rel="stylesheet" href="lib/css/chevrons.css">
<link rel="stylesheet" href="lib/css/tabs.css">
<link rel="stylesheet" href="lib/css/content.css">
<link rel="stylesheet" href="lib/css/buttons.css">
<link rel="stylesheet" href="lib/css/forms.css">
<link rel="stylesheet" href="lib/css/lists.css">
<link rel="stylesheet" href="lib/css/spinners.css">
<link rel="stylesheet" href="lib/css/icomoon.css">
<link rel="stylesheet" href="lib/css/stack.css">
<link rel="stylesheet" href="lib/css/sliders.css">
```

Figura 36 – Referências aos arquivos CSS utilizados

```
<script src="lib/js/dialogs.js"></script>
<script src="lib/js/forms.js"></script>
<script src="lib/js/toasts.js"></script>
<script src="lib/js/utils.js"></script>
<script src="lib/js/action-bars.js"></script>
<script src="lib/js/spinners.js"></script>
<script src="lib/js/tabs.js"></script>
<script src="lib/js/SocialSharing.js"></script>
<script type="text/javascript" src="cordova.js"></script>
<script type="text/javascript" src="lib/js/index.js"></script>
```

Figura 37 - Referências aos arquivos JavaScript utilizados

Após referenciados estes arquivos, os componentes do *Fries* podem ser utilizados. Para a inclusão de uma barra de cabeçalho foi utilizada a classe *"action-bar"*, descrita no arquivo *"action-bars.js"*, como mostrados na *Figura 38.*

```
<header class="action-bar"> <!-- Cabeçalho de título -->
        <a href="javascript: void(0);" class="app-icon action">
                <img src="lib/icons/icon-multimetro2.png">
        </a>
        <h1 class="title">Multímetro</h1>
</header>
```

Figura 38 – Comandos para inclusão de barra de cabeçalho

Para a mudança entre o contexto de exibição dos valores medidos continuamente para a exibição do gráfico, foram utilizadas *tabs*, que permitem a navegação em múltiplas telas. Também foram usadas *tabs* para a seleção da medição a ser realizada. Para cada tipo de medida um caractere definido é enviado via *Bluetooth* para a placa

fazendo com que o microcontrolador efetue a leitura dos sinais no borne referente a leitura de interesse. Este elemento descrito no arquivo *"tabs.js"* evidencia de forma visual bastante clara e seleção ativa, neste caso na área de leitura ou gráfico, ou no tipo de medição sendo realizada. A *Figuras 39 e 40* mostram as linhas de código utilizadas.

```
<nav class="tab-fixed">
        <ul class="tab-inner">
                <li class="active"><a href="#item1" data-
ignore="true">Leitura</a></li> <!-- Aba de 'Leitura' -->
                <li><a href="#item2" data-
ignore="true">Gráfico</a></li>
        </ul>
</nav>
```

Figura 39 – Comandos para a criação das *Tabs*

```
<nav class="tab-fixed">
        <ul class="tab-inner">
                <li><a href=" javascript:
bluetoothSerial.write('T');" class="action" title="Tensão"
id="TButton"><img src="lib/icons/icon-tensao.png"></a></li>
<!-- Botão para seleção de Voltímetro -->
                <li><a href=" javascript:
bluetoothSerial.write('C');" class="action" title="Corrente"
id="CButton"><img src="lib/icons/icon-corrente.png"></a></li>
<!-- Botão para seleção de Amperímetro -->
                <li><a href=" javascript:
bluetoothSerial.write('R');" class="action" title="Resistência"
id="RButton"><img src="lib/icons/icon-resistencia.png"></a></li>
<!-- Botão para seleção de ohmímetro -->
        </ul>
</nav>
```

Figura 40 – Lista de conteúdo de cada *Tab*

Ao clicar no botão *"Conectar"*, é chamada, dentre várias, a função *"bluetoothSerial.connect"*, *Figura 41*, que é responsável pela requisição de conexão com o dispositivo selecionado, e em caso positivo chama a função *"app.connect"*, que redireciona a aplicação para uma segunda tela que contém as medições e controles, e em caso negativo chama a função *"app.disconnect"*, que exibe uma mensagem informando o usuário de que a conexão com o dispositivo falhou.

```
bluetoothSerial.connect(device,app.onconnect,app.ondisconnect);
```

Figura 41 – Comando para estabelecimento de conexão *Bluetooth*

Os dados recebidos via *Bluetooth* são alocados na variável *"message"* e exibidos na página HTML em uma *div* chamada *"messages"*. Dependendo do tipo de medida realizada é exibido um texto contendo a respectiva unidade. A *Figura 42* mostra as linhas de código utilizadas.

```
onmessage: function(message) {
        if (unidade == "T"){
                ext = " V";
                messages.innerHTML = message + ext;    // Exibe na
div 'messages' a mensagem recebida por Bluetooth
                messages_.innerHTML = message + ext;    // Exibe na
div 'messages' a mensagem recebida por Bluetooth
        }
        if (unidade == "C"){
                ext = " mA";
                messages.innerHTML = message + ext;    // Exibe na
div 'messages' a mensagem recebida por Bluetooth
                messages_.innerHTML = message + ext;    // Exibe na
div 'messages' a mensagem recebida por Bluetooth
        }
        if (unidade == "R"){
                ext = " Ohms";
                messages.innerHTML = message + ext;    // Exibe na
div 'messages' a mensagem recebida por Bluetooth
                messages_.innerHTML = message + ext;    // Exibe na
div 'messages' a mensagem recebida por Bluetooth
        }
        medicao = message;
},
```

Figura 42 – Atribuições em variáveis das medições realizadas

As principais funções para o armazenamento dos valores medidos são descritas em seguida e mostradas na *Figura 43*.

A função *"gotFS"* é responsável pela criação do arquivo gerando seu nome baseado no horário em que o usuário faz a requisição de gravação. Já a função *"gotFileWriter"* grava no arquivo gerado os valores armazenados na variável *"data"* nomeando-a com o tipo de medida realizada, dentre tensão, corrente ou resistência.

Nesta mesma função, é feito o compartilhamento do arquivo por *"window.plugins.socialsharing.share"*.

```
function gotFS(fileSystem) {
        var today = new Date();
        var dd = today.getDate();
        var mm = today.getMonth() + 1; //Janeiro é 0
        var yyyy = today.getFullYear();
        var hour = today.getHours();
        var min = today.getMinutes();
        var sec = today.getSeconds();

        top.nomearq = 'Download/' +
yyyy+'.'+mm+'.'+dd+'.'+hour+'.'+min+'.'+sec+'.'+'multimetro_log.tx
t';

        fileSystem.root.getFile(nomearq, {create: true},
gotFileEntry, fail);
        window.plugins.socialsharing.share(null, null,
'file:///storage/sdcard0/'+nomearq);
}

function gotFileEntry(fileEntry) {
        fileEntry.createWriter(gotFileWriter, fail);
}

function gotFileWriter(writer) {
        writer.onwrite = function(evt) {
                console.log("Salvo com sucesso");
        };
        writer.seek(writer.length);

        writer.write(unidade + ' = [' + data + ']');
        writer.abort();
}
```

Figura 43 – Funções para armazenamento em arquivo das medições realizadas

O cálculo dos valores máximo e mínimo, *Figura 44*, atingidos pela variável de interesse, tensão ou corrente, é feito com o uso de uma variável auxiliar, neste caso chamada *ordenado*, que possui o mesmo valor da variável *data*, que armazena os valores medidos para exibição no gráfico. A necessidade do uso de um vetor auxiliar com os mesmos valores se dá pela utilização da função *sort()*, que ordena o vetor de forma crescente. Caso o vetor *data* fosse ordenado de forma direta, o gráfico mostraria uma reta crescente, então para manter este vetor intacto foi utilizada uma variável auxiliar que pode ser manipulada sem prejuízo. Ordenado em ordem crescente, a

primeira posição do vetor representa o valor mínimo, e a última posição o valor máximo.

```
ordenado.sort();
min_ = ordenado[0];
max_ = ordenado[ordenado.length - 1];

minimo.innerHTML = "Mínimo: " + min_ + ext;
maximo.innerHTML = "Máximo: " + max_ + ext;
```

Figura 44 – Obtenção dos valores máximo e mínimo

O gráfico é gerado a partir de um vetor contendo leituras recebidas via *Bluetooth*. Este vetor é atualizado continuamente por meio do deslocamento dos valores do fim do vetor até a posição inicial, a cada novo recebimento o valor presente na primeira posição do vetor é apagada, dando espaço para a seguinte, e assim por diante, até que a última posição seja liberada, para armazenar um novo valor. Após este armazenamento é chamada a função responsável pela plotagem do gráfico baseado neste vetor de medidas. Encerrado esse processo, ele é realizado novamente de forma recursiva. Um recurso presente na biblioteca utilizada para geração do gráfico altera a faixa de valores do eixo y com a finalidade de permitir uma melhor resolução do sinal apresentado na tela. A *Figura 45* mostra as linhas de código utilizadas.

```
$(function () {
        var totalPoints = 500;

        function getData() {
                if (data.length > 0) {
                        data = data.slice(1);
                        ordenado = ordenado.slice(1);
                }
                while (data.length < totalPoints) {
                        data.push(parseFloat(medicao));
//Preenche na última posição do vetor 'data' o valor da variável
'medicao'
                        testearq.push(parseFloat(medicao));
                        ordenado.push(parseFloat(medicao));
                }   //funções .push(Preenche na última posição do
vetor) e parseFloat(converte string para float) descritas no
jquery.js
                // Agrupa os valores de y com os valores de x
                var res = [];
                for (var i = 0; i < data.length; ++i)
                        res.push([i, data[i]])
                return res;
        }
        var updateInterval = 1;

        // setup plot
        var options = {
                series: { shadowSize: 0 }, // plota o gráfico sem
sombra para melhor desempenho
                xaxis: { min: 0, max: 500, tickFormatter:
function (t) { return t + " ms"; } },
                yaxis: { min: 0,
                        panRange: [0, 50]
                },
                pan: {
                        interactive: true
                }
        };

        var plot = $.plot($("#placeholder"), [ getData() ],
options); //placeholder é a div que corresponde ao gráfico

        function update() {
                plot = $.plot($("#placeholder"), [ getData() ],
options); //placeholder é a div que corresponde ao gráfico

                plot.setData([ getData() ]);  // Função
plot.setData descrita em jquery.flot.js e atribui valores ao
gráfico
                plot.draw();    // Função plot.draw descrita em
jquery.flot.js e atribui valores ao gráfico
                setTimeout(update, updateInterval); //A cada
estouro da variável updateInterval a função update é chamada, e o
gráfico é atualizado
```

Figura 45 – Geração do gráfico com as medições realizadas

O diagrama da *Figura 46* mostra o fluxo de execução do programa desenvolvido para o *Android*. Os códigos em HTML, CSS e *JavaScript* podem ser vistos nos *Apêndices C, D e E*, respectivamente.

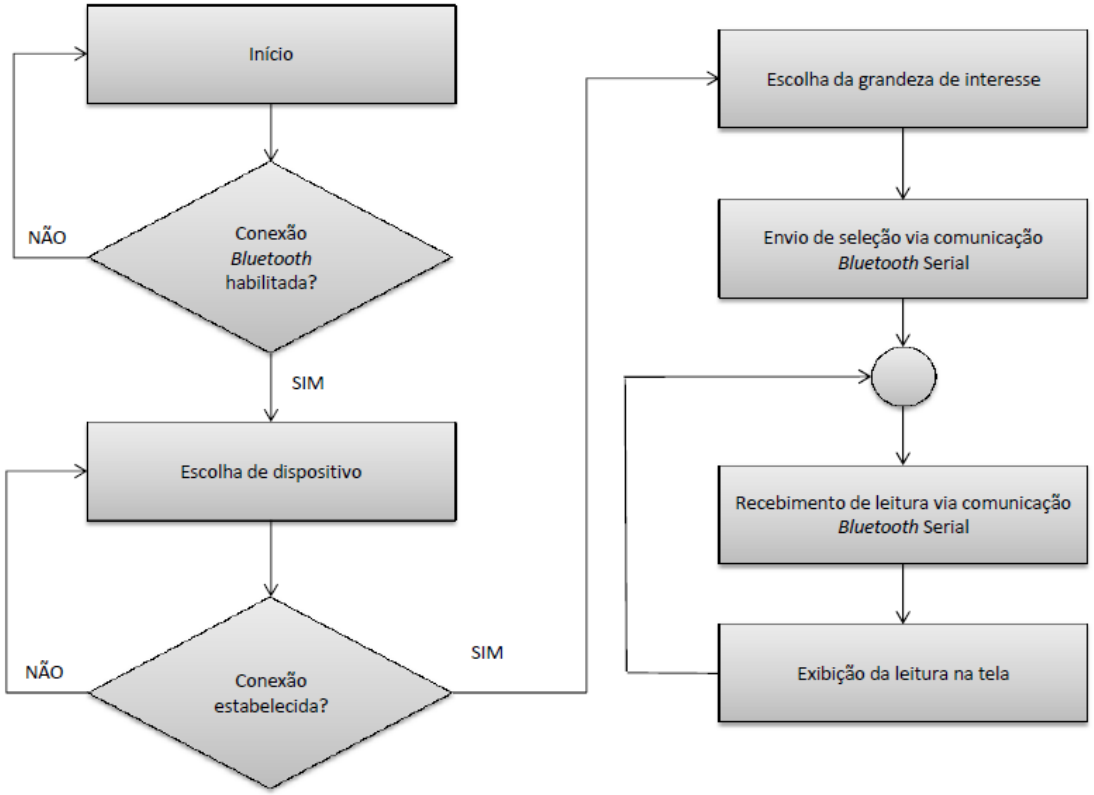


Figura 46 - Fluxograma do programa para *Android*

Capítulo 4

Resultados

4.1 *Hardware*

Foram realizados testes em bancada a fim de validar o circuito e o programa projetados. Para comparação foi utilizado um multímetro comercial U1231A, fabricado pela *Agilent Technologies*. Este multímetro possui suporte ao *software Agilent GUI Data Logger*, distribuído pela própria *Agilent Technologies* em seu site oficial [21].

Abaixo são listadas algumas características do *Hardware* desenvolvido final:

- *Hardware* baseado no microcontrolador ATMega328P, *Atmel Corporation;*
- Faixa de medição de tensão: 0 V – 30 V;
- Faixa de medição de corrente: 0 mA – 500 mA;
- Faixa de medição de resistência: 0 Ω – 1,023 M Ω;
- Tensão de operação: 5,8 V – 18,3 V;
- Dimensões da placa: 80 mm x 50 mm;
- Consumo de potência médio: 167,55mW;
- Autonomia de bateria, para o uso de uma bateria de 9 V de 280 mAh: 23 horas e 10 minutos aproximadamente.

Com o uso deste *software*, foram gravadas as medições realizadas para fim de comparação. Tal equipamento não possui suporte para medição direta de corrente, assim foi utilizado um resistor de potência de valor 1 Ω e 1% de tolerância e medido o valor de tensão sobre ele, dessa forma através da lei de Ohm, $V = R.I$, têm-se que $I = \frac{V}{R}$, assim para a resistência de 1Ω, $I = \frac{V}{1} = V$.

Utilizando o recurso de gravação de medições realizadas em um intervalo de tempo na aplicação desenvolvida para plataformas *Google Android*, podem-se comparar de forma simples as leituras com as realizadas através do uso do multímetro comercial.

Tendo em mãos os arquivos gerados na medição com o uso de ambos dispositivos, foram realizadas análises visuais. Tendo sido plotados dois gráficos, um contendo curvas superpostas, e outro mostrando a diferença entre as duas curvas, pode-se analisar tanto a forma como cada equipamento realizou a medida, quanto a diferença, que permite calcular de forma prática o erro máximo relativo obtido entre as medidas.

4.1.1 Medida de tensão

Com o uso do circuito desenvolvido e do multímetro comercial, foi medida a queda de tensão sobre um dos braços de um potenciômetro, utilizado como divisor resistivo, com o objetivo de a partir de uma fonte de 12 V criar diferentes níveis de tensão para teste. A medição ocorreu conforme *Figura 47*, que exibe a conexão em paralelo dos dois instrumentos, que fazem a aquisição da grandeza elétrica no mesmo instante.

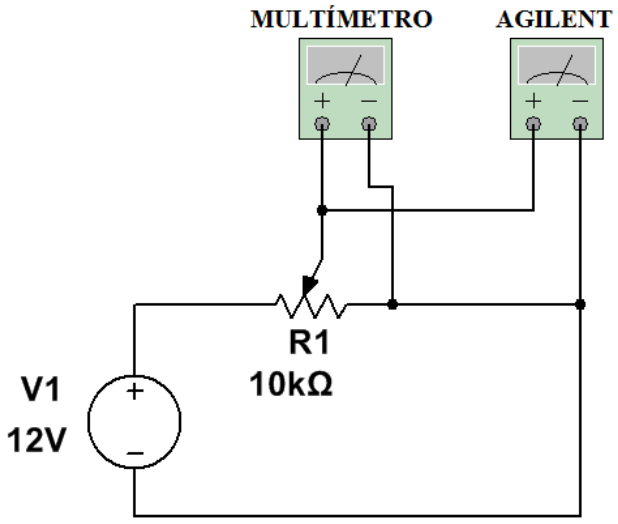


Figura 47 - Circuito para medição de tensão

Posteriormente, com todos os valores armazenados em computador. foi gerado um gráfico contendo todas as medições superpostas, da forma que se pode observar de forma clara a resposta de cada instrumento. A *Figura 48* exibe este gráfico.

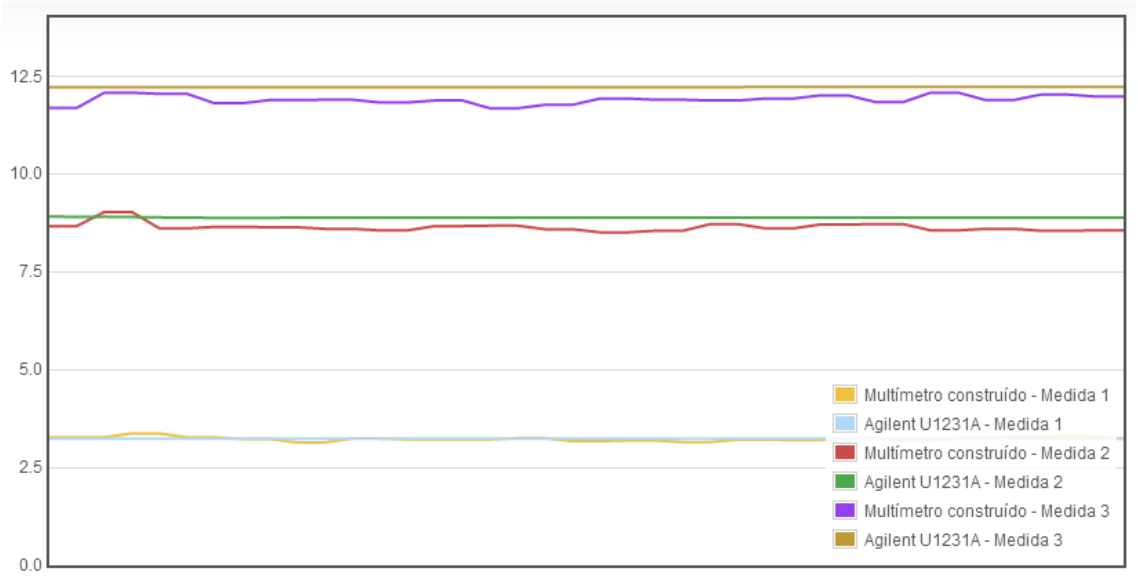


Figura 48 - Comparação de medidas de tensão

Gráficos representando a diferença entre leituras de cada instrumento são vistos a seguir para cada valor de tensão medido, e a partir destes valores foi possível calcular o erro relativo baseado no valor do instrumento comercial.

A *Figura 49* exibe a diferença entre as medições. Foi calculado o erro relativo máximo com base nos valores apresentados na janela de medição obtida e um valor médio de 3,24 V, valor de referência obtido com o uso do multímetro comercial.

$$Erro_{relativo\ máximo} = \frac{0{,}14\ V}{3{,}24\ V} = 4{,}32\%$$

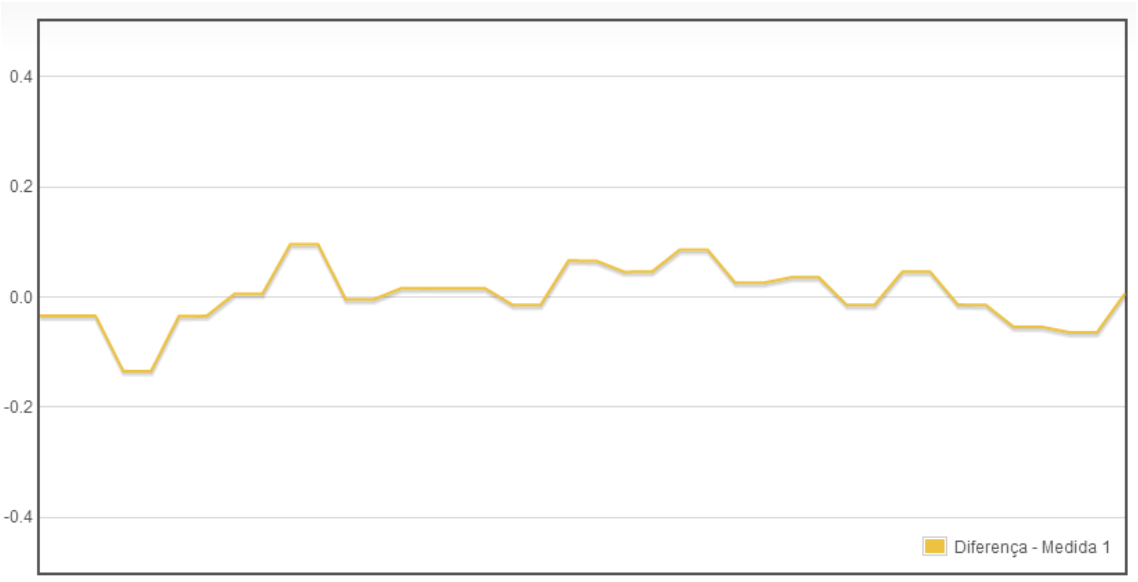


Figura 49 - Diferença entre a Medida 1 (Tensão)

O mesmo procedimento foi realizado para um valor de referência médio de 8,89V e a diferença mostrada na *Figura 50*.

$$Erro_{relativo\ máximo} = \frac{0{,}38\ V}{8{,}89\ V} = 4{,}27\%$$

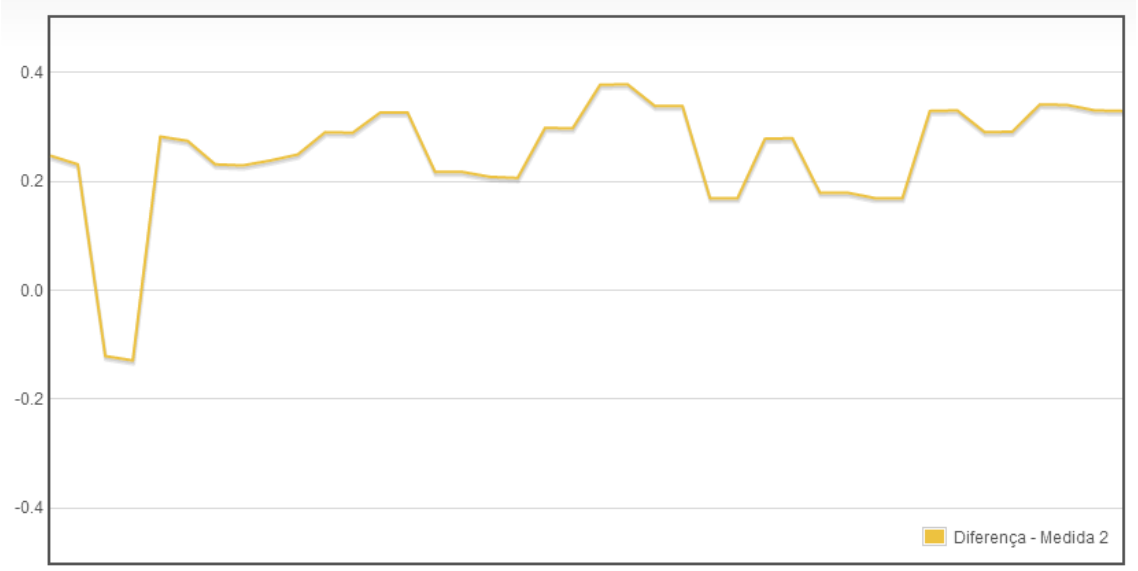


Figura 50 - Diferença entre a Medida 2 (Tensão)

Por fim, com uma ultima referência de 12,22V e valores de diferença representados na *Figura 51* foi calculado um ultimo valor de erro relativo experimental.

$$Erro_{relativo\ máximo} = \frac{0{,}54\ V}{12{,}22\ V} = 4{,}42\%$$

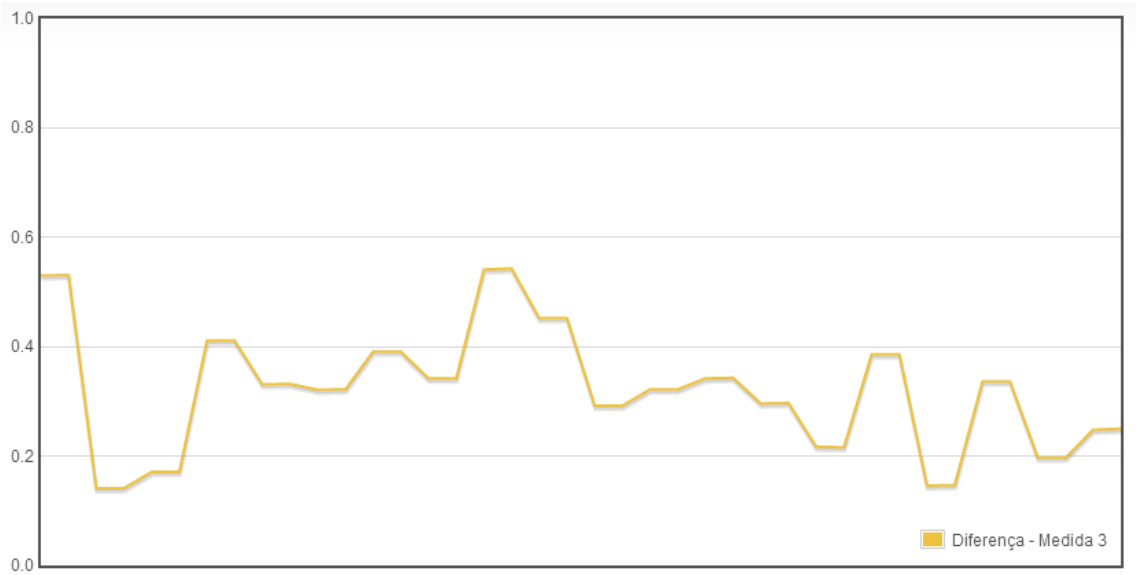


Figura 51 - Diferença entre a Medida 3 (Tensão)

Como se pode observar, todos os valores de erro relativo obtidos se mostraram bem próximos e com valor entre 4,27% e 4,42%. Comparando os resultados entre os instrumentos, nota-se uma boa resposta tendo em vista a simplicidade e baixo custo do circuito empregado para a medição de tensão elétrica no circuito desenvolvido e a qualidade do instrumento comercial de referência, cuja calibração garante uma medição mais precisa.

4.1.2 Medida de corrente

Para teste de medição de corrente elétrica, foram feitas aquisições de dados para três valores de corrente que atravessam um conjunto de resistores alimentados por uma fonte de tensão. Foi utilizada uma fonte de tensão de 3,3 V e resistores de 5% de tolerância nos valores de 33 Ω, 50 Ω e 100 Ω. Como o multímetro *Agilent U1231A* não possui suporte nativo a medição de corrente foi então utilizado um resistor de 1 Ω em série ao circuito, e medida sua queda de tensão. Através deste valor foi obtida a corrente. O circuito contendo a conexão dos instrumentos é o apresentado abaixo na *Figura 52*.

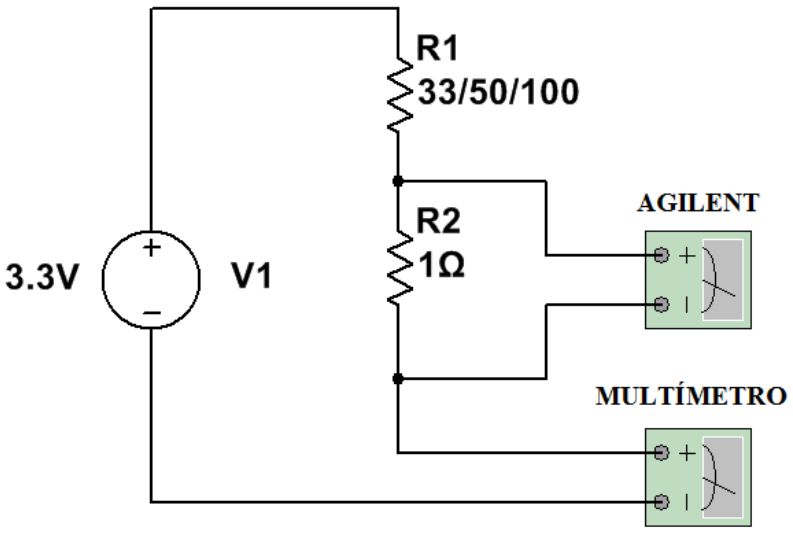


Figura 52 - Circuito para medição de corrente

Os valores de corrente esperados estão listados na *Tabela 2*:

Tabela 2 - Valores de corrente esperados

Medida	Resistor	Corrente
1	34 Ω	97,06 mA
2	51 Ω	64,71 mA
3	101 Ω	32,67 mA

A *Figura 53* exibe seis curvas sobrepostas, cada qual contendo as medições de ambos os instrumentos para cada caso durante uma janela de tempo.

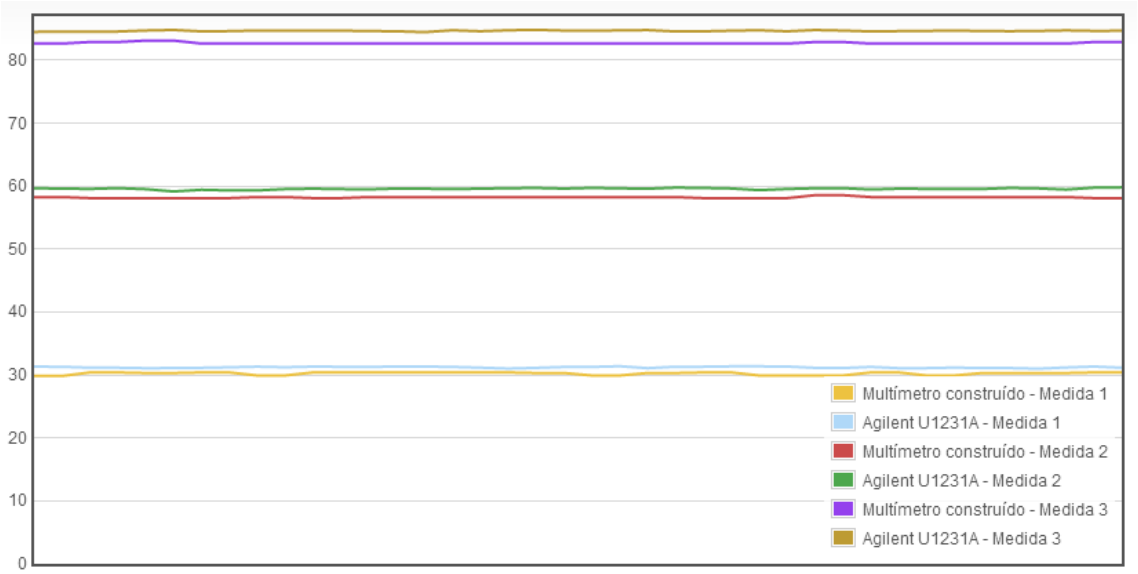


Figura 53 - Comparação de medidas de corrente

Foram gerados gráficos contendo a diferença entre as medições de cada aparelho com o objetivo de se calcular o erro relativo, dado que será utilizado para avaliar a precisão de medição do dispositivo desenvolvido se comparado a um equipamento comercial de qualidade e robustez que possui certificado de calibração garantindo confiabilidade nas medições.

A *Figura 54* exibe a diferença entre os valores medidos com ambos instrumentos com o objetivo de comparação e cálculo do erro relativo máximo experimental, também exibido abaixo.

$$Erro_{relativo\ máximo} = \frac{1{,}48\ mA}{31{,}22\ mA} = 4{,}74\%$$

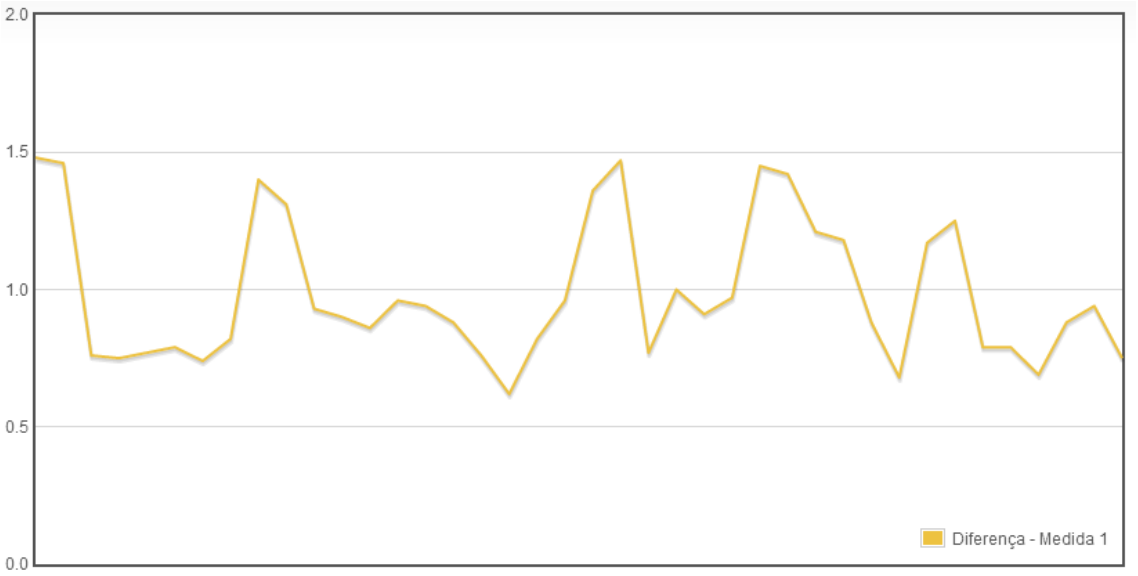


Figura 54 - Diferença entre a Medida 1 (Corrente)

Um segundo conjunto de medições de corrente foi adquirido, assim com a sua diferença que pode ser vista na *Figura 55*. Novamente foi calculado o erro relativo máximo experimental.

$$Erro_{relativo\ máximo} = \frac{1{,}71\ mA}{59{,}58\ mA} = 2{,}87\%$$

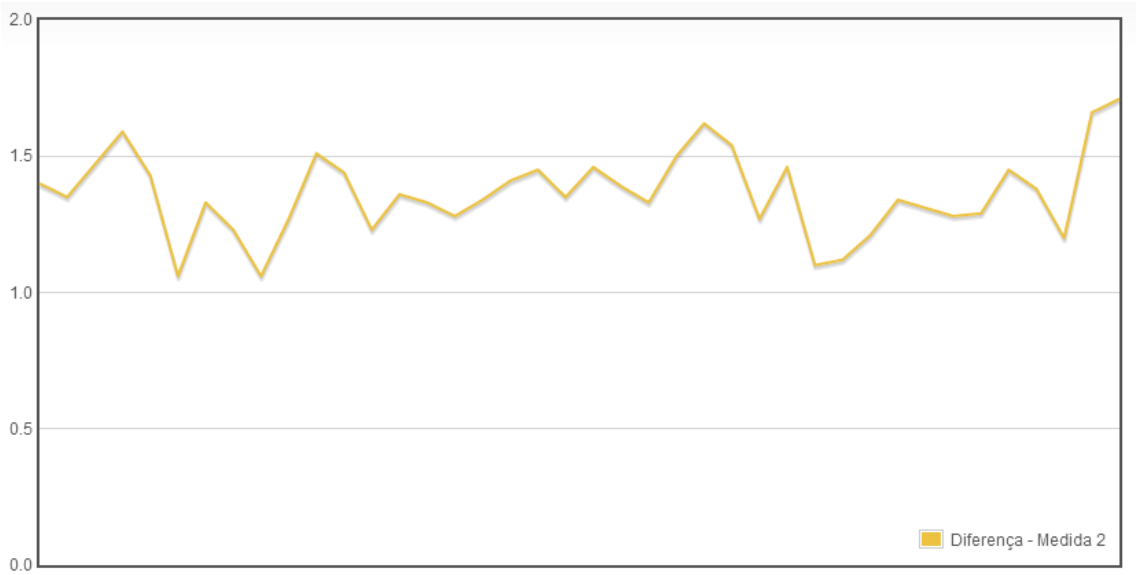


Figura 55 - Diferença entre a Medida 2 (Corrente)

A ultima série de valores de corrente foi adquirida para calculo do erro relativo máximo. A *Figura 56* mostra o gráfico que representa a diferença entre a medição dos instrumentos.

$$Erro_{relativo\ máximo} = \frac{2{,}19\ mA}{84{,}66\ mA} = 2{,}59\%$$

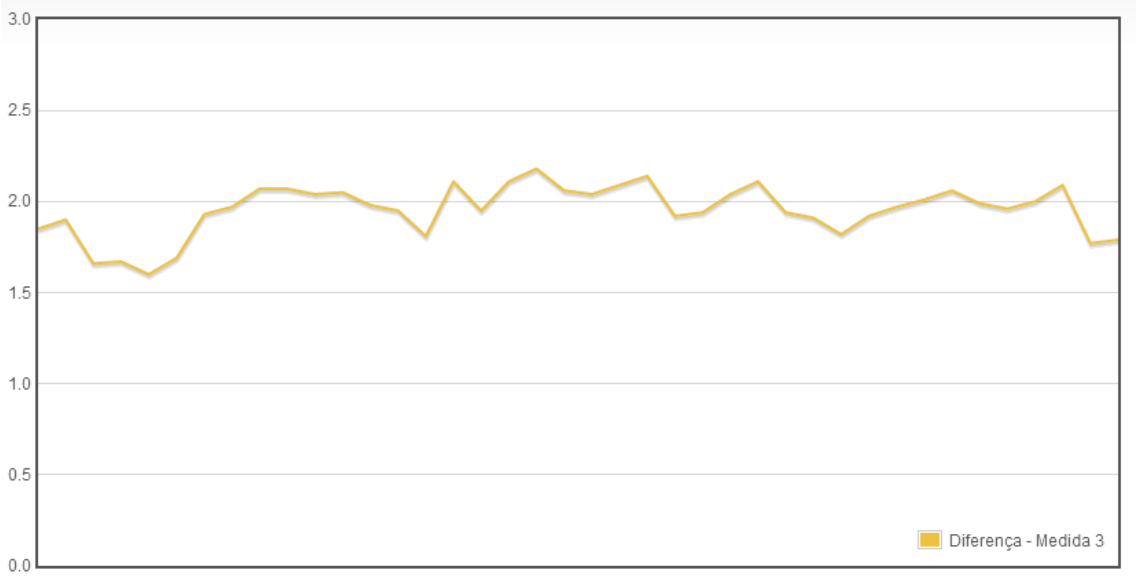


Figura 56 - Diferença entre a Medida 3 (Corrente)

O maior erro relativo experimental calculado foi da ordem de 4,74% para o valor mais baixo de corrente medida. Com o aumento do valor de corrente, observa-se uma diminuição do erro relativo, sendo observado o valor e 2,59%. Os valores obtidos se mostraram bastante satisfatórios se comparados a um equipamento comercial específico de grande precisão e confiabilidade. Desta forma, pode-se constatar a viabilidade da solução proposta para a medição de corrente elétrica.

4.1.3 Medida de resistência

Sendo a resistência elétrica uma grandeza que sofre pouca variação ao longo do tempo em um ambiente controlado, cuja temperatura permanece quase constante, a comparação se mostra mais simples. A *Figura 57* demonstra a conexão do instrumento de medição, seja ele o multímetro desenvolvido, ou o multímetro de referência *Agilent U1231A*, com o resistor que se deseja aferir.

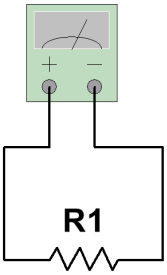


Figura 57 - Circuito para medição de resistência

A *Tabela 3* exibe valores medidos no instrumento desenvolvido e no instrumento comercial de referência, para resistores de 100 Ω, 1 kΩ, 10 kΩ, 100 kΩ e 500 kΩ de 5% de tolerância. A tabela ainda contém o erro relativo percentual das medições realizadas pelo dispositivo construído.

Tabela 3 - Valores medidos de resistência

Valor ideal	Multímetro construído	Agilent U1231A	Erro relativo (%)
56 Ω	53,87 Ω	55,3 Ω	2,585895118
100 Ω	98,82 Ω	99,3 Ω	0,483383686
1 kΩ	980,23 Ω	980 Ω	0,023469388
10 kΩ	9,97789 kΩ	9,91 kΩ	0,68506559
100 kΩ	99,88189 kΩ	99,1 kΩ	0,788990918
500 kΩ	503,34959 kΩ	499,2 kΩ	0,831247997

Nota-se um valor de erro relativo máximo igual a aproximadamente 2,59% para um valor baixo de resistência elétrica dada a faixa de medição do circuito. Este valor se reduz para menos de 1% para valores maiores de resistência. Experimentalmente, estes valores demonstram um grande resultado na medição de valores de resistência comparado a um multímetro comercial completo e de qualidade.

4.2 *Software*

O *software* desenvolvido para o microcontrolador operou como o desejado. A comunicação com o módulo *Bluetooth* não apresentou nenhum tipo de problema e a comunicação entre dispositivos *Google Android* e a placa desenvolvida se mostrou bastante eficiente.

A conversão analógico/digital dado cada tipo de medição de interesse também operou como o esperado e o uso da tensão de referência interna de 1,1 V foi bastante efetivo no cálculo das tensões de entrada para obtenção de medições mais precisas.

Para o *software* do *smartphone*, a usabilidade por meio dos comandos em tela opera como o esperado. Dentre os possíveis problemas previstos na execução da aplicação dois casos foram tratados. O primeiro consistia no caso de, por algum motivo, seja falta de alimentação ou alcance, o *hardware* desenvolvido perdesse comunicação com o dispositivo *Android*, neste caso a aplicação retorna a tela inicial de escolha do periférico *Bluetooth* com o qual se deseja conectar, bastando ao usuário entrar novamente com a opção desejada. O outro caso previsto consistia na ausência de memória no dispositivo que executa a aplicação para o armazenamento das medições realizadas; Este caso foi contornado por meio de uma função de erro presente no *framework PhoneGap* que alerta ao usuário a incapacidade da gravação de medidas.

Capítulo 5

Conclusão

O trabalho proposto consistiu da criação de um dispositivo que ampliasse as capacidades e recursos já existentes de aparelhos *smartphones* e *tablets*, utilizando suas capacidades de processamento, memória, e principalmente neste caso, a tela, para medidas de grandezas elétricas de tensão, corrente e resistência.

Empregando protocolos e padrões de comunicação suportados pela grande maioria dos aparelhos portáteis mais atuais, é possível criar soluções que atendam a necessidade de muitos usuários, com grande robustez e capacidades que seriam muito caras se fossem implementadas unicamente através de componentes e circuitos integrados dedicados em um *hardware*, como por exemplo, o acesso à rede móvel de telefonia e acesso remoto a internet. Outro recurso presente nas plataformas móveis com sistemas operacionais complexos é a imensa disponibilidade de produtos e serviços existentes, como por exemplo, o armazenamento de dados em serviços *online*, compartilhamento de mensagem via correio eletrônico, além de uma série de outros que aumentam as possibilidades de criação e desenvolvimento de projetos.

A aplicação desenvolvida se resume em um instrumento para medição de grandezas elétricas, sendo elas, tensão, corrente e resistência, que se conecta a aparelhos portáteis com sistema operacional *Google Android* e, por meio de um aplicativo criado para tal propósito, é operado. Para a comunicação entre os dispositivos foi utilizado o protocolo sem fio *Bluetooth*, que se mostrou uma excelente solução, já que seu suporte em dispositivos móveis é bastante presente, incrementa um baixo consumo de energia quando habilitado, além do módulo em *hardware* necessário para a adição deste periférico em um sistema embarcado possuir preço bastante acessível.

Todo o *software* foi desenvolvido com o uso de ferramentas abertas e recursos que permitiam o acesso a alguns periféricos e recursos dos aparelhos de sistema *Android*, como a comunicação *Bluetooth* e armazenamento local de informação. A criação da aplicação foi realizada com *framework PhoneGap* que proporcionou o acesso a estes recursos e facilitou o projeto de interface com usuário por meio do suporte a

linguagens de programação para *web*, como o HTML, que permite a simples manipulação de recursos visuais e simples desenvolvimento em computadores.

O *hardware* construído foi baseado na plataforma *Arduino*, que agiu como grande facilitadora no desenvolvimento da aplicação embarcada para o microcontrolador empregado, já que a prototipagem é rápida devido a interface de desenvolvimento acessível, além de meios de programação. O microcontrolador empregado, o ATmega328P, fabricado pela *Atmel Corporation*, possui todos os periféricos necessários para a aplicação desenvolvida, assim como recursos de processamento e memória. Era desejada uma solução em *hardware* simples e barata, além de desejado um equipamento de dimensões reduzidas, já que se tratava de uma aplicação de caráter móvel, assim toda a parte de condicionamento de sinal foi desenvolvida de forma a atender a estas necessidades.

Com o objetivo de analisar o *hardware* desenvolvido, foi utilizado um multímetro comercial de qualidade e garantia de medições precisas através de certificação de calibração, o aparelho em questão foi um *Agilent U1231A*. Este equipamento possui suporte a conectividade com computadores e *software* para aquisição de medições, que foi um grande facilitador para a observação dos resultados. Assim, com as medições armazenadas pelo *software* móvel e as obtidas pela ferramenta *Agilent GUI Data Logger* o equipamento desenvolvido pode ser analisado em suas capacidades de medição de grandezas elétricas. Observa-se uma melhor resposta a medição de valores de resistência elétrica, em que foi obtido um erro máximo relativo de aproximadamente 2,59%, comparado as medições realizadas pelo instrumento de referência, um excelente resultado para a solução proposta. Já a medição de tensão gerou valores de erro relativo próximos a 4%, este resultado também se mostra positivo, levando novamente em questão o custo e simplicidade da solução de instrumentação encontrada. Por fim, analisa-se a medição de corrente elétrica que apresentou valores de erro menores que 4,74%, chegando a um mínimo de 2,59%. Estes valores se mostram melhores se comparados aos obtidos na medição de valores de tensão elétrica.

Comparando com o instrumento comercial o erro relativo máximo observado não ultrapassou o valor de 5%. Este valor se mostra bastante positivo comparado com o multímetro comercial em questão, o *Agilent U1231A*, uma vez que este equipamento é bastante robusto e confiável. Assim a criação de um dispositivo que atenda as necessidades de comunicação, consumo de bateria e tamanho, além de custo, se mostrou viável. Enfim, o projeto desenvolvido operou conforme projetado e esperado, e o

conjunto de *software* para plataformas móveis em conjunto com uma extensão em *hardware* se mostrou algo muito interessante e que permite a criação de uma série de aplicações.

A quantidade de conhecimento obtido e o grande número de problemas enfrentados durante todo o projeto foram de grande valia para formação acadêmica, uma vez que tornou necessário o aprofundamento e estudo em diversas áreas, sendo as principais: instrumentação eletrônica e sistemas digitais.

6.1 Trabalhos Futuros

Sendo a plataforma *PhoneGap* compatível com sistema *Apple iOS*, assim como o *plugin Bluetooth Serial* utilizado, a substituição do módulo de *hardware Bluetooth* empregado por um com suporte a tecnologia *Bluetooth Low Energy* permitiria a migração da aplicação desenvolvida para esta plataforma, já que restringe a comunicação com periféricos a esta tecnologia, aumentando a quantidade de sistemas e equipamentos suportados.

Outro ponto com relação a comunicação via protocolo *Bluetooth* é o estudo de viabilidade da conexão com múltiplas placas, para assim realizar medições distintas ao mesmo tempo.

Pode-se considerar também a mudança no *hardware* para permitir a medição de tensão e corrente alternada da rede possibilitando até mesmo o cálculo do fator de potência.

Outra alteração que pode ser feira ao projeto é a utilização de apenas dois bornes para a conexão de ponte de prova, em que a placa identificasse automaticamente a medição realizada após a seleção feita via *software*, trazendo maior facilidade e comodidade ao usuário.

Referências Bibliográficas

[1] Richter, E. Multímetro interfaceado de baixo custo para aquisição de dados. Química Nova, v. 27, n. 2, p.313, 2004

[2] Datasheet. Atmel AVR ATmega328P. Disponível em: <http://www.atmel.com/Images/doc8161.pdf> Acesso em 15 abr.2014

[3] Arduino. Disponível em: <http://arduino.cc//> Acesso em 15 abr.2014

[4] Processing. Disponível em: <http://www.processing.org/> Acesso em 15 abr.2014

[5] Android. Disponível em: <http://www.android.com/> Acesso em 15 abr.2014

[6] Android SDK. Disponível em: <http://developer.android.com/sdk/index.html> Acesso em 15 abr.2014

[7] PhoneGap. Disponível em: < http://phonegap.com/> Acesso em 15 abr.2014

[8] Bluetooth Serial Plugin for Phonegap. Disponível em: <https://github.com/don/BluetoothSerial> Acesso em 15 abr.2014

[9] PhoneGap Social Sharing Plugin. Disponível em: <https://github.com/EddyVerbruggen/SocialSharing-PhoneGap-Plugin> Acesso em 15 abr.2014

[10] Fries. Disponível em: < http://getfri.es/> Acesso em 15 abr.2014

[11] Linux Mint. Disponível em: <http://www.linuxmint.com/> Acesso em 15 abr.2014

[12] USBtinyISP. Disponível em: <https://learn.adafruit.com/usbtinyisp> Acesso em 15 abr.2014

[13] Current Sense Circuit Collection, Linear Technology, 2005. Disponível em: <http://cds.linear.com/docs/en/application-note/an105.pdf> Acesso em 15 abr.2014

[14] CADSoft Eagle PCB Design. Disponível em: <http://www.cadsoftusa.com/> Acesso em 15 abr.2014

[15] SketchUp. Disponível em: <http://www.sketchup.com/> Acesso em 15 abr.2014

[16] Node Package Modules. Disponível em: <http://www.npmjs.org/> Acesso em 15 abr.2014

[17] Node.js. Disponível em: <http://www.nodejs.org/> Acesso em 15 abr.2014

[18] Flot. Disponível em: <http://www.flotcharts.org/> Acesso em 15 abr.2014

[19] jQuery. Disponível em: <http://www.jquery.com/> Acesso em 15 abr.2014

[20] jQuery Mobile. Disponível em: <http://www.jquerymobile.com/> Acesso em 15 abr.2014

[21] Agilent GUI Data Logger. Disponível em: <http://www.home.agilent.com/agilent/software.jspx?ckey=878442&lc=por&cc=BR&nid=-11143.0.00&id=878442> Acesso em 30 abr.2014

[22] Cengage Learning, Inc.Handheld multimeter offers wireless data streming. Wireless design & development, v. 18, n. 1, p.29, 2010

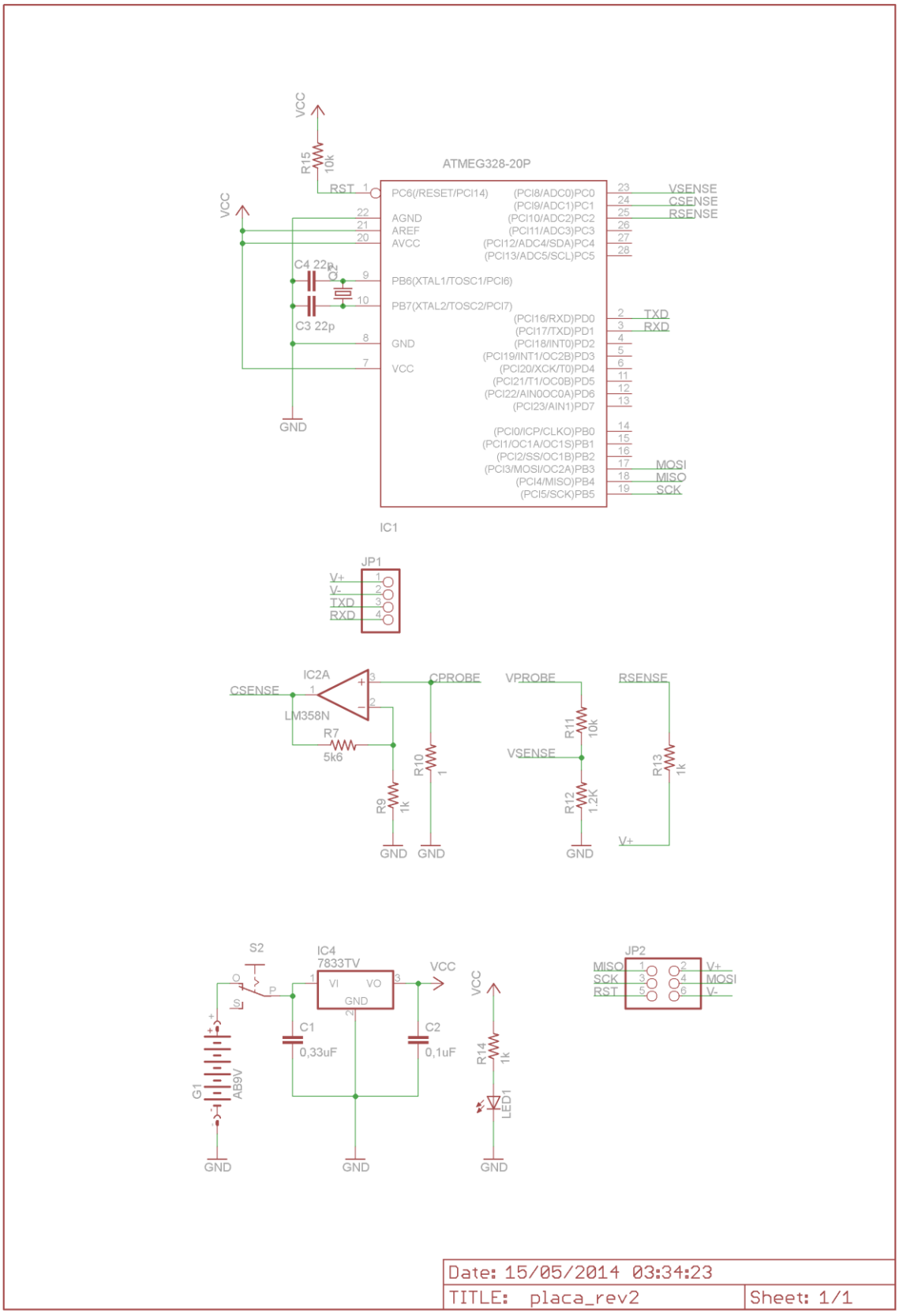
[23] Lecklider, T. Handheld DMM features count. EE, evaluation engineering, v. 51, n. 11, p.22, 2012

[24] Grenez, F. Wireless prototype based on pressure and bending sensors for measuring gait corrected quality. Sensors, v. 13, n. 8, p. 9679 - 9703, 2013

[25] Saquib et al ; Amin, I. Wireless Control of Miniaturized Mobile Vehicle for Indoor Surveillance. IOP Conference Series: Materials Science and Engineering, v. 51, n. 1, p. 12025, 2013

Apêndices

Apêndice A – *Hardware* Desenvolvido



Apêndice B – Código do Arduino

```
const unsigned int tensaoIn = A0; // Porta de entrada do conversor AD para medição de tensão
const unsigned int correnteIn = A1; // Porta de entrada do conversor AD para medição de corrente
const unsigned int resistenciaIn = A2; // Porta de entrada do conversor AD para medição de resistência
char modo = 'T';

long Vref() {
  //Usa um circuito de referência interna do microcontrolador
  //para fazer medições mais precisas

  long resultado;
  // Lê valor de 1.1V de referência reference contra AVcc
  ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  delay(2); // Espera Vref estabilizar
  ADCSRA |= _BV(ADSC); // Conversão
  while (bit_is_set(ADCSRA,ADSC));
  resultado = ADCL;
  resultado |= ADCH<<8;
  resultado = 1125300L / resultado; // Recalcula AVcc em escala de mV
  return resultado;  // Retorna o valor de referência
}

void setup(){
  Serial.begin(38400);  //Seta o baudrate do módulo Bluetooth;
}

void loop(){

  float tensao;
  float corrente;
  float resistencia;

  if (Serial.available() > 0) {
    modo = Serial.read();
  }

  if (modo == 'T')  // Modo voltímetro
  {
    unsigned int sinalADv;
    double tensao;
    double x;
    x = Vref() / 1000.0;  // Calcula o valor de referência para medição da tensão
    sinalADv = analogRead(tensaoIn);  // Lê o valor de tensão presente na porta A0
    tensao = (sinalADv / 1023.0) * x / 0.107; // Vsense ~= Vprobe*0.107;
    Serial.println(tensao);  // Envia o valor de tensão medido
  }
  if (modo == 'C')  // Modo amperímetro
  {
    unsigned int sinalADc;
    double corrente;
    double x;

    x = Vref() / 1000.0;  // Calcula o valor de referência para medição da corrente
    sinalADc = analogRead(correnteIn);  // Lê o valor de tensão presente na porta A1
    corrente = (sinalADc / 1023.0) * x;
    corrente = corrente / 6.6;
    corrente = corrente * 1000.0;
    Serial.println(corrente);  // Envia o valor de corrente medido
  }
  if (modo == 'R')  // Modo ohmímetro
  {
    double sinalADr;
```

```
    double x;
    double resistencia;

    x = Vref() / 1000.0;  // Calcula o valor de referência para medição da resistência
    sinalADr = analogRead(resistenciaIn);  // Lê o valor de tensão presente na porta A2
    sinalADr = (sinalADr / 1023.0) * x;
    resistencia = sinalADr / (x - sinalADr);
    resistencia = resistencia * 1000.0;
    Serial.println(resistencia);  // Envia o valor de resistência medido
  }
}
```

Apêndice C – Código HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <meta name = "format-detection" content = "telephone=no"/>
        <meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-
scale=1, minimum-scale=1, width=device-width;" />
        <link rel="stylesheet" type="text/css" href="lib/css/topcoat-mobile-
light.min.css" />
        <link rel="stylesheet" type="text/css" href="lib/css/index.css" />
        <link rel="stylesheet" href="lib/css/base.css">
        <link rel="stylesheet" href="lib/css/action-bars.css">
        <link rel="stylesheet" href="lib/css/chevrons.css">
        <link rel="stylesheet" href="lib/css/tabs.css">
        <link rel="stylesheet" href="lib/css/content.css">
        <link rel="stylesheet" href="lib/css/buttons.css">
        <link rel="stylesheet" href="lib/css/forms.css">
        <link rel="stylesheet" href="lib/css/lists.css">
        <link rel="stylesheet" href="lib/css/spinners.css">
        <link rel="stylesheet" href="lib/css/icomoon.css">
        <link rel="stylesheet" href="lib/css/stack.css">
        <link rel="stylesheet" href="lib/css/sliders.css">
        <script language="javascript" type="text/javascript"
src="lib/js/flot/jquery.js"></script>
        <script language="javascript" type="text/javascript"
src="lib/js/flot/jquery.flot.js"></script>
        <title>Multímetro</title>
    </head>

    <body ontouchstart="">

        <header class="action-bar"> <!-- Cabeçalho de título -->
            <a href="javascript: void(0);" class="app-icon action">
                <img src="lib/icons/icon-multimetro2.png">
            </a>
            <h1 class="title">Multímetro</h1>
        </header>


        <div id="content">

            <div id="connection">
                <div style="margin: 60px"><br><br><br>
                    <br>Dispositivo: <br/>
                    <select id="deviceList">
                        <option>Descobrindo...</option>
                    </select>
                </div>
                <p>
                    <a class="btn" id="connectButton">Conectar</a>
                </p>
                <p>
                    <br>
                    <a class="btn" id="listButton">Listar Dispositivos</a>
                </p>
            </div>

            <div id="chat" style="margin-top: 45px">

                <nav class="tab-fixed">
                    <ul class="tab-inner">
                        <li class="active"><a href="#item1" data-
ignore="true">Leitura</a></li> <!-- Aba de 'Leitura' -->
                        <li><a href="#item2" data-ignore="true">Gráfico</a></li> <!--
Aba de 'Gráfico' -->
                    </ul>
                </nav>
```

```
                <header class="content" style="margin-top:40px">
                    <div class="slider">
                        <ul>
                            <li id="item1" class="tab-item active">
                                <ul class="inset">
                                    <div id="messages" style="font-size:43px;
color:AntiqueWhite2"></div><br>
                                    <p>
                                        <div id="maximo" style="text-align:center; font-
size:22px; color:LightSkyBlue"></div>
                                        <div id="minimo" style="text-align:center; font-
size:22px; color:LightSkyBlue"></div>
                                    </p>
                                    <br>
                                    <p id="stringResponseText"> </p>
                                </p>
                                <!--<nav class="action-bar fixed-bottom">
                                <ul class="actions flex">  -->
                                    <nav class="tab-fixed" style="position:absolute;
top:429px">
                                        <ul class="tab-inner">
                                            <li><a href=" javascript:
bluetoothSerial.write('T');" class="action" title="Tensão" id="TButton"><img
src="lib/icons/icon-tensao.png"></a></li>    <!-- Botão para seleção de Voltímetro -->
                                            <li><a href=" javascript:
bluetoothSerial.write('C');" class="action" title="Corrente" id="CButton"><img
src="lib/icons/icon-corrente.png"></a></li>    <!-- Botão para seleção de Amperímetro --
>
                                            <li><a href=" javascript:
bluetoothSerial.write('R');" class="action" title="Resistência" id="RButton"><img
src="lib/icons/icon-resistencia.png"></a></li>  <!-- Botão para seleção de ohmímetro -->
                                        </ul>
                                    </nav>
                                </ul>
                            </li>
                            <li id="item2" class="tab-item" style="text-align:center">
                                <div class="inset"><br>
                                    <div id="messages_" style="font-size:23px"></div>
                                    <div id="message_res" style="font-
size:23px"><p>Opção não disponível</p></div>
                                    <div id="placeholder"
style="width:310px;height:335px"></div>
                                    <p><br>
                                        <a class="btn" id="gravaButton">Gravar
Medidas</a>
                                    </p>
                                </div>
                            </li>
                        </ul>
                    </div>
                </header>

            </div>
        </div>

        <footer>
            <div id="statusMessage" style="text-align:center"></div>
        </footer>

        <script src="lib/js/stack.js"></script>
        <script src="lib/js/dialogs.js"></script>
        <script src="lib/js/forms.js"></script>
        <script src="lib/js/toasts.js"></script>
        <script src="lib/js/utils.js"></script>
        <script src="lib/js/action-bars.js"></script>
        <script src="lib/js/spinners.js"></script>
        <script src="lib/js/tabs.js"></script>
        <script src="lib/js/SocialSharing.js"></script>
        <script type="text/javascript" src="cordova.js"></script>
        <script type="text/javascript" src="lib/js/index.js"></script>

        <script type="text/javascript">
            app.initialize();
        </script>

        <script type="text/javascript">
```

```
        $(function () {
            var totalPoints = 500;

            function getData() {
                if (data.length > 0) {
                    data = data.slice(1);
                    ordenado = ordenado.slice(1);
                }

                while (data.length < totalPoints) {
                    data.push(parseFloat(medicao));   //Preenche na última posição do
vetor 'data' o valor da variável 'medicao'
                    testearq.push(parseFloat(medicao));
                    ordenado.push(parseFloat(medicao));
                }   //funções .push(Preenche na última posição do vetor) e
parseFloat(converte string para float) descritas no jquery.js
                // Agrupa os valores de y com os valores de x
                var res = [];
                for (var i = 0; i < data.length; ++i)
                    res.push([i, data[i]])
                return res;
            }

            var updateInterval = 1;    //uma amostra a cada 10 milisegundos

            // setup plot
            var options = {
                series: { shadowSize: 0 }, // plota o gráfico sem sombra para melhor
desempenho
                xaxis: { min: 0, max: 500, tickFormatter: function (t) { return t + "
ms"; } },
                yaxis: { min: 0,
                    panRange: [0, 50]
                },
                pan: {
                    interactive: true
                }
            };

            var plot = $.plot($("#placeholder"), [ getData() ], options); //placeholder
é a div que corresponde ao gráfico

            function update() {
                ordenado.sort();
                min_ = ordenado[1];
                max_ = ordenado[ordenado.length - 1];

                minimo.innerHTML = "Mínimo: " + min_ + ext;
                maximo.innerHTML = "Máximo: " + max_ + ext;

                min_ = 0;
                max_ = 0;

                plot = $.plot($("#placeholder"), [ getData() ], options); //placeholder
é a div que corresponde ao gráfico

                plot.setData([ getData() ]);  // Função plot.setData descrita em
jquery.flot.js e atribui valores ao gráfico
                // since the axes don't change, we don't need to call plot.setupGrid()
                plot.draw();    // Função plot.draw descrita em jquery.flot.js e atribui
valores ao gráfico
                setTimeout(update, updateInterval); //A cada estouro da variável
updateInterval a função update é chamada, e o gráfico é atualizado
            }
            update();
        });
    </script>

    </body>
</html>
```

Apêndice D – Código CSS

```
.fadein {
  opacity: 1;
  -webkit-transition: opacity 1s ease-in;
}

.fadeout {
  opacity: 0;
  -webkit-transition: opacity 1s ease-out;
}

textarea {
    font: 16px "Source Sans", helvetica, arial, sans-serif;
    font-weight: 200;
    display: block;
    -webkit-border-radius: 6px;
    width: 100%;
    height: 140px;
}

#connection {
    margin-bottom: 40px;
    text-align:center;
}

#chat {
    display:none;

}


#sendButton {
    margin-top: 4px;
}

#messages {
    text-align:center;
    margin-top: 40%;
    font-size: 40px;
}

select {
    min-width: 200px;
}

form {
    margin-bottom: 20px;
}

footer {
    margin: 20px;
}
```

Apêndice E – Código JAVA

```
'use strict';

top.medicao = '0.0';
top.unidade = "T";
top.testearq = [];
top.data = [];
top.ordenado = [];
top.ext = " V"

var app = {
    initialize: function() {
        this.bind();
        listButton.style.display = "none";  // Na inicialização do programa esconde o
botão de listar dispositivos
    },
    bind: function() {
        document.addEventListener('deviceready', this.deviceready, false);
    },
    deviceready: function() {

        // Determina botões para as funções
        connectButton.ontouchstart = app.connect;
        listButton.ontouchstart = app.list;

        TButton.ontouchstart = app.tensao;  // Ao pressionar o botão 'TButton' chama a
função 'tensao'
        CButton.ontouchstart = app.corrente;    // Ao pressionar o botão 'CButton' chama
a função 'corrente'
        RButton.ontouchstart = app.resistencia; // Ao pressionar o botão 'RButton' chama
a função 'resistencia'
        gravaButton.ontouchstart = app.grava;

        // Aguardando mensagens
        bluetoothSerial.subscribe("\n", app.onmessage,
app.generateFailureFunction("Subscribe Failed"));

        // recebe a lista de dispositivos
        setTimeout(app.list, 2000);
    },
    list: function(event) {
        deviceList.firstChild.innerHTML = "Descobrindo...";
        app.setStatus("Procurando por dispositivos Bluetooth...");  // Imprime na tela
mensagem de status
        bluetoothSerial.list(app.ondevicelist, app.generateFailureFunction("Listagem
Falhou")); // Função para listar dispositivos, em caso positivo chama a função
'ondevicelist', caso contrário gera função de erro
    },
    connect: function() {
        var device = deviceList[deviceList.selectedIndex].value;
        app.disable(connectButton);
        app.setStatus("Conectando..."); // Imprime na tela mensagem de status
        console.log("Requisitando conexão com " + device);  // Imprime mensagem de log
no console(não visível na tela)
        bluetoothSerial.connect(device, app.onconnect, app.ondisconnect);   // Tenta
conectar com dispositivo selecionado, em caso positivo chama a função app.connect, em
caso negativo chama a função app.disconnect
    },
    disconnect: function(event) {
        if (event) {
            event.preventDefault();
        }

        app.setStatus("Desconectando...");  // Imprime na tela mensagem de status
        bluetoothSerial.disconnect(app.ondisconnect); //Ao desconectar o dispositivo
chama a função 'ondisconnect'
    },
    ondevicelist: function(devices) {
        var option;

        // Remove os dispositivos existentes
```

```
        deviceList.innerHTML = "";
        app.setStatus("");

        devices.forEach(function(device) {  // Indexa os dispositivos Bluetooth
encontrados

            option = document.createElement('option');
            if (device.hasOwnProperty("uuid")) {
                option.value = device.uuid;
            } else if (device.hasOwnProperty("address")) {
                option.value = device.address;
            } else {
                option.value = "ERROR " + JSON.stringify(device);
            }
            option.innerHTML = device.name;
            deviceList.appendChild(option);
        });

        if (devices.length === 0) {

            option = document.createElement('option');
            option.innerHTML = "Sem dispositivos Bluetooth";
            deviceList.appendChild(option);

            if (cordova.platformId === "ios") { // Dispositivo BLE(Bluetooth Low Energy)
                app.setStatus("Nenhum dispositivo Bluetooth encontrado.");  // Imprime
na tela mensagem de status
            } else { // Dispositivo Android
                app.setStatus("Por favor pareie com um dispositivo Bluetooth.");    //
Imprime na tela mensagem de status
            }

            app.disable(connectButton); // Desabilita o estado do botão 'connectButton'
            listButton.style.display = "";
        } else {
            app.enable(connectButton);
            listButton.style.display = "none"; // Esconde o botão de listar dispositivos
            app.setStatus("Encontrado" + (devices.length === 1 ? " " : "s ") +
devices.length + " dispositivo" + (devices.length === 1 ? "." : "s."));  // Imprime na
tela mensagem de status
        }

    },
    onconnect: function() {
        connection.style.display = "none";  // Esconde a div 'connection'
        chat.style.display = "block";   // Exibe a div 'chat'
        message_res.style.display = "none";  // Esconde mensagem alerta ohmímetro
        app.setStatus("Conectado"); // Imprime na tela mensagem de status
    },
    ondisconnect: function(reason) {
        var details = "";
        if (reason) {
            details += ": " + JSON.stringify(reason);
        }
        connection.style.display = "block"; // Exibe a div 'connection'
        app.enable(connectButton);
        chat.style.display = "none";    // Esconde a div 'chat'
        app.setStatus("Desconectado");  // Imprime na tela mensagem de status
    },
    onmessage: function(message) {
        if (unidade == "T"){
            ext = " V";
            messages.innerHTML = message + ext;   // Exibe na div 'messages' a mensagem
recebida por Bluetooth
            messages_.innerHTML = message + ext;   // Exibe na div 'messages' a mensagem
recebida por Bluetooth
        }
        if (unidade == "C"){
            ext = " mA";
            messages.innerHTML = message + ext;   // Exibe na div 'messages' a mensagem
recebida por Bluetooth
            messages_.innerHTML = message + ext;   // Exibe na div 'messages' a mensagem
recebida por Bluetooth
        }
        if (unidade == "R"){
            ext = " Ohms";
```

```
            messages.innerHTML = message + ext;   // Exibe na div 'messages' a mensagem
recebida por Bluetooth
            messages_.innerHTML = message + ext;   // Exibe na div 'messages' a mensagem
recebida por Bluetooth
        }
            medicao = message;
    },
    setStatus: function(message) {
        console.log(message);

        window.clearTimeout(app.statusTimeout);
        statusMessage.innerHTML = message;  // Exibe na div 'statusMessage' a mensagem
        statusMessage.className = 'fadein'; // Animação de 'fadein' na exibição da
mensagem

        // Automaticamente apaga uma mensagem de status após determinado tempo
        app.statusTimeout = setTimeout(function () {
            statusMessage.className = 'fadeout';    // Animação de 'fadeout' na exibição
da mensagem
        }, 5000);
    },
    enable: function(button) {
        button.className = button.className.replace(/\bis-disabled\b/g,''); // Chama o
estado de habilitação de botões
    },
    disable: function(button) {
        if (!button.className.match(/is-disabled/)) {   // Checa se o botão está
habilitado
            button.className += " is-disabled"; // Chama o estado de desabilitação de
botões
        }
    },
    tensao: function() {
        bluetoothSerial.write("T"); // Envia o caracter 'T' via Bluetooth
        unidade = "T";
        for (var i = data.length - 1; i >= 0; i--) {
            data[i] = 0;
            testearq[i] = 0;
            ordenado[i] = 0;
        };
        placeholder.style.display = "block";  // Exibe o gráfico
        maximo.style.display = "block";  // Exibe o valor máximo
        minimo.style.display = "block";  // Exibe o valor minímo
        gravaButton.style.display = "block";  // Exibe o botão para gravar medições
        messages_.style.display = "block";  // Exibe medições na tab de gráfico
        message_res.style.display = "none";  // Esconde mensagem alerta ohmímetro
    },
    corrente: function() {
        bluetoothSerial.write("C"); // Envia o caracter 'C' via Bluetooth
        unidade = "C";
        for (var i = data.length - 1; i >= 0; i--) {
            data[i] = 0;
            testearq[i] = 0;
            ordenado[i] = 0;
        };
        placeholder.style.display = "block";  // Exibe o gráfico
        maximo.style.display = "block";  // Exibe o valor máximo
        minimo.style.display = "block";  // Exibe o valor minímo
        gravaButton.style.display = "block";  // Exibe o botão para gravar medições
        messages_.style.display = "block";  // Exibe medições na tab de gráfico
        message_res.style.display = "none";  // Esconde mensagem alerta ohmímetro
    },
    resistencia: function() {
        bluetoothSerial.write("R"); // Envia o caracter 'R' via Bluetooth
        unidade = "R";
        for (var i = data.length - 1; i >= 0; i--) {
            data[i] = 0;
            testearq[i] = 0;
            ordenado[i] = 0;
        };
        placeholder.style.display = "none";  // Esconde o gráfico
        maximo.style.display = "none";  // Esconde o valor máximo
        minimo.style.display = "none";  // Esconde o valor minímo
        gravaButton.style.display = "none";  // Esconde o botão para gravar medições
        messages_.style.display = "none";  // Esconde medições na tab de gráfico
        message_res.style.display = "block";  // Exibe mensagem alerta ohmímetro
    },
```

```
    grava: function() {
        document.addEventListener("deviceready", onDeviceReady, false);

        function onDeviceReady() {
            window.requestFileSystem(LocalFileSystem.PERSISTENT, 1024 * 1024, gotFS,
fail);
        }

        function gotFS(fileSystem) {
            var today = new Date();
            var dd = today.getDate();
            var mm = today.getMonth() + 1; //Janeiro é 0
            var yyyy = today.getFullYear();
            var hour = today.getHours();
            var min = today.getMinutes();
            var sec = today.getSeconds();

            top.nomearq = 'Download/' +
yyyy+'.'+mm+'.'+dd+'.'+hour+'.'+min+'.'+sec+'.'+'multimetro_log.txt';

            fileSystem.root.getFile(nomearq, {create: true}, gotFileEntry, fail);
            window.plugins.socialsharing.share(null, null,
'file:///storage/sdcard0/'+nomearq);
        }

        function gotFileEntry(fileEntry) {
            fileEntry.createWriter(gotFileWriter, fail);
        }

        function gotFileWriter(writer) {
            writer.onwrite = function(evt) {
                console.log("Salvo com sucesso");
            };
            writer.seek(writer.length);

            writer.write(unidade + ' = [' + testearq + ']');
            writer.abort();
        }

        function fail(error) {
            console.log("error : "+error.code);
        }
    },
    generateFailureFunction: function(message) {
        var func = function(reason) {
            var details = "";
            if (reason) {
                details += ": " + JSON.stringify(reason);
            }
            app.setStatus(message + details);
        };
        return func;
    }
};
```

Apêndice F – Custo de Materiais

Custo de projeto estimado em dólares cotados nas lojas Newark (http://www.newark.com/) e Dealextreme (http://www.dx.com/).

Quantidade	Componente	PREÇO
1	Microcontrolador ATmega328P	$ 3,30
2	Resistor Axial 10 kΩ 1% tolerância	$ 0,03
2	Resistor Axial 1 kΩ 1% tolerância	$ 0,03
1	Resistor Axial 1,2 kΩ 1% tolerância	$ 0,03
1	Resistor Axial 5,6 kΩ 1% tolerância	$ 0,03
1	Resistor Axial 1 Ω 0,5W 1% tolerância	$ 0,04
1	Resistor Axial 56 Ω 1% tolerância	$ 0,03
2	Capacitor cerâmico 22 pF 50V	$ 0,03
1	Capacitor cerâmico 0,1 µF 50V	$ 0,16
1	Capacitor cerâmico 0,33 µF 50V	$ 0,85
1	Regulador de tensão LM7833	$ 0,60
1	Cristal oscilador de 16 MHz	$ 0,31
1	Amplificador Operacional LM358N	$ 0,48
1	LED 5mm vermelho	$ 0,12
1	Barra de pinos	$ 0,04
1	Clip para bateria de 9 volts	$ 0,39
1	Módulo Bluetooth	$ 7,36
1	Placa de circuito impresso	$ 1,40
1	Carcaça de acrílico	$ 3,00
1	Par de pontas de prova	$ 1,20
3	Borne vermelho para pino banana	$ 0,51
1	Borne preto para pino banana	$ 0,51
1	Chave 3 terminais 90 graus	$ 0,13
1	Soquete para CI 8 pinos	$ 0,17
1	Soquete para CI 28 pinos	$ 0,45

PREÇO FINAL	$ 22,25