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

Desenvolvimento de sistema para aquisição e acesso remoto a medidas elétricas por meio de dispositivos móveis e internet

Aluno: André Aureliano Biagioni
Professor Orientador: Evandro L. L. Rodrigues

SÃO CARLOS
2014

ANDRÉ AURELIANO BIAGIONI

Desenvolvimento de sistema para aquisição e acesso remoto a medidas elétricas por meio de dispositivos móveis e internet

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

Biagioni, André Aureliano
B576d Desenvolvimento de sistema para aquisição e acesso remoto a medidas elétricas por meio de dispositivos móveis e internet / André Aureliano Biagioni; orientador Evandro Luís 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. Arduino. 3. Bluetooth. 4. Google Android. 5. Web. I. Título.

FOLHA DE APROVAÇÃO

Nome: André Aureliano Biagioni

Título: "Desenvolvimento de sistema para aquisição e acesso remoto a medidas elétricas por meio de dispositivos móveis e internet"

Trabalho de Conclusão de Curso defendido e aprovado em 28 / 11 / 2014,

com NOTA 10,0 (dez , zero), **pela Comissão Julgadora:**

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

Prof. Dr. Maximiliam Luppe - (SEL/EESC/USP)

Prof. Dr. Marcelo Andrade da Costa Vieira - (SEL/EESC/USP)

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

Agradecimentos

Agradeço aos meus pais, meus maiores professores, por terem permitido e proporcionado tudo o que era necessário para a minha formação.

Agradeço aos meus irmãos, Armando e Bruna, pelo companheirismo e amizade, persistentes de forma inabalável até hoje.

Agradeço aos queridos Acássio Roque, André Grana, Bruno Callegaro, Filipe Bonatti, Letícia Myazato, Priscila dos Reis, Rafael Ferrassini e Stephania Salvato pela amizade sincera nos tempos de São Carlos.

Agradeço ao Prof. Evandro pelas inúmeras conversas e conselhos durante toda a graduação.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

ADC	*Analog to Digital Converter*
AC	*Alternating Current*
ADK	*Accessory Development Kit*
API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
CSV	*Comma-separated values*
DC	*Direct Current*
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
HTML	*HyperText Markup Language*
I2C	*Inter Intergrated Circuit*
IDE	*Integrated Development Environment*
LCD	*Liquid Crystal Display*
NPM	*Node Package Module*
OLED	*Organic Light-Emitting Diode*
PWM	*Pulse-Width Modulation*
RAM	*Random Access Memory*
RMS	*Root Mean Square*
SDK	*Software Development Kit*
SRAM	*Static Random-Access Memory*
UART	*Universal Asynchronous Receiver/Transmitter*
USB	*Universal Serial Bus*

Resumo

Este trabalho descreve o desenvolvimento de uma plataforma para aquisição de medidas elétricas de tensão e corrente alternada, para cálculo de potência consumida por uma carga, por meio de um dispositivo responsável pelas leituras e envio dos dados obtidos via comunicação sem fio *Bluetooth* a um dispositivo móvel *Google Android*, que, além de exibir os valores medidos em sua tela em tempo real de forma gráfica e textual, envia as leituras para um banco de dados, para que sejam acessadas remotamente via página *web* por um usuário.

Palavras-chave: Instrumentação, *Arduino*, *Bluetooth*, *Google Android*, *WEB*

Abstract

This work describes the development of a platform for acquiring electrical measurements of voltage and alternating current, for calculating power consumed by a load, by means of a device responsible for reading and sending the data via Bluetooth wireless communication to a mobile Google Android, which, in addition to displaying the values measured on your screen in real-time graphical and textual form, sends the readings to a database, to be accessed remotely via a web page by a user.

Keywords: Instrumentation, *Arduino*, *Bluetooth*, *Google Android*, *WEB*.

1. Introdução

A aplicação de instrumentação na análise de circuitos elétricos e eletrônicos é algo comum, de grande uso, e de extrema necessidade para quem tem contato direto na área de engenharia. No entanto, dispositivos que permitem a leitura simplificada de medidas de interesse que permeiam o cotidiano das pessoas tem se tornado algo comum e que tem recebido bastante foco de grandes fabricantes, que passaram a realizar pesquisas e disponibilizar equipamentos e recursos para que usuários comuns tenham controle sobre informações que para eles anteriormente eram intangíveis.

Um exemplo disto são os medidores inteligentes, dispositivos capazes de gravar o consumo de energia elétrica, água ou gás, durante um intervalo de tempo e enviar esta informação até uma base para a cobrança de consumo. Este tipo de tecnologia dispensa a necessidade de funcionários para visitas periódicas com a finalidade de ler o consumo em um medidor, além de realizar medições mais precisas e ao longo do tempo, permitindo o acesso e análise dos dados em gráficos. Tal recurso pode ser utilizado de forma auxiliar e promover o consumo consciente, resultando em quedas na demanda de energia.

Como base pode ser tomado um estudo [1] realizado pela *CenterPoint Energy Inc.* e o Departamento de Energia americano que determinou que 71% dos consumidores pesquisados alteraram sua forma de consumo de energia como resultado do acesso e conhecimento das medições por meio de visualizadores internos na residência, de forma que passaram a economizar. Outro exemplo que pode ser tomado é o de um teste [2] realizado em uma pequena cidade no estado de Iowa, Estados Unidos, em que 1000 residências receberam medidores inteligentes, que reportavam o consumo de energia a cada 15 minutos e enviavam a um *data center* para análise. As leituras realizadas poderiam ser vistas pelos consumidores em um *website*, onde teriam acesso a outras informações, como consumos passados e comparações de consumo com outras residências que possuíam características similares. O resultado do teste apontou uma redução de onze por cento no consumo de energia elétrica dentre os meses de julho e dezembro de 2010.

Da mesma forma que pode ser feita a análise do consumo total de uma residência, a medição em cargas pontuais podem auxiliar no controle e análise de consumo, permitindo apontamentos, como a detecção de equipamentos com consumo

acima do esperado, dadas a suas características, ou ocorrência de algum problema ou defeito. Este tipo de equipamento pode ser facilmente encontrado no mercado, porém a grande maioria apresenta suas leituras em um *display* gráfico digital, ou cursor analógico, não permitindo o registro para posterior análise.

Assim, foi proposto o desenvolvimento e a construção de um dispositivo para a medição de tensão e corrente alternada, que permitisse leituras de consumo de potência de uma carga remotamente, por meio de um navegador de internet. O envio destas leituras para uma base de dados é feita por intermédio de um *smartphone* com sistema *Google Android [3]*, o qual é conectado ao dispositivo construído por meio do protocolo sem fio *Bluetooth*.

1.1 Motivação

Com a existência de soluções comerciais de medidores de potência para uso doméstico, que apresentam suas leituras em displays gráficos, além dos medidores de energia inteligentes, mostra-se interessante o desenvolvimento de plataformas que permitam a comunicação e acesso a dados aquisicionados por aparelhos e instrumentos, para registro, análise e processamento computacional, assim como o acesso remoto. Outro fator de interesse é a criação de um circuito barato, de forma a ser uma alternativa aos circuitos comerciais, e facilmente replicável, fazendo o uso de recursos de comunição de aparelhos móveis, como os *smartphones*, sem a necessidade do uso dispositivos de difícil acesso e complexidade para esta função.

1.2 Objetivos

São listados abaixo os principais objetivos do trabalho realizado:

- Desenvolvimento de projeto e construção de um dispositivo de baixo custo para medição de grandezas de tensão e corrente alternada, com comunicação com *smartphones* por meio de protocolo *Bluetooth*;
- Desenvolvimento de aplicação para *smartphone* com sistema *Google Android*, para aquisição e visualização das medidas por meio de comunicação sem fio *Bluetooth*, e envio para banco de dados com o uso de internet;

- Desenvolvimento de página *web* para acesso e visualização das medidas realizadas.

1.3 Estruturação do Trabalho

O trabalho aqui descrito é dividido em cinco partes, segmentados como mostrado abaixo:

1. *Conceituação Teórica*: apresentação de conceitos e conhecimentos de suma importância para o desenvolvimento do trabalho;
2. *Descrição do projeto e materiais e métodos utilizados*: explanação dos materiais e métodos empregados na construção e desenvolvimento do trabalho
3. *Implementação e Desenvolvimento*: descrição do projeto desenvolvido, com detalhamento de conceituação e aplicação.
4. *Resultados*: apresentação dos resultados encontrados com o *hardware* desenvolvido, sendo apresentados os testes realizados.
5. *Conclusão*: apontamentos finais e validação dos objetivos propostos no trabalho.

2. Conceituação Teórica

Aqui são descritos, de forma geral, os principais pontos entorno da realização do projeto, sendo eles a Instrumentação eletrônica, Microcontroladores e recursos computacionais.

2.1 Instrumentação Eletrônica

A instrumentação eletrônica consiste do estudo e desenvolvimento de técnicas para a medição de grandezas elétricas por meio de instrumentos que não afetem, ou que causem o mínimo de impacto, a dinâmica original do circuito no qual se deseja realizar alguma medição.

Este tipo de equipamento pode ser analógico ou digital, sendo o primeiro comumente baseado em instrumentos de bobina móvel e imã permanente (IBMIP), em que uma agulha desloca sobre um mostrador com valores de referência marcados em uma graduação, dada a corrente que atravessa a bobina ao qual está fixada. Sendo este instrumento por natureza um amperímetro para a medição de outras grandezas, como tensão, é necessário o uso de circuitos condicionadores de sinal, que convertem valores de tensão em corrente para a excitação da bobina responsável por deslocar o ponteiro do instrumento de medição. Já os instrumentos digitais são construídos principalmente com o emprego de circuitos conversores analógico digitais (ADC, do inglês *analog to digital converter*), sendo este, diferente do IBMIP, um medidor de tensão por natureza. Assim para a medição de grandezas elétricas, estas devem ser condicionadas em valores de tensão. Os valores obtidos na conversão geram palavras digitais que são processadas e exibidas em mostradores gráficos, como *displays* LCD (*Liquid Crystal Display*), possuindo como vantagem sobre os instrumentos analógicos a maior facilidade de leitura das medições realizadas, além de recursos como armazenamento de um valor medido.

2.1.1 Conversão Analógico-Digital

Por meio da conversão analógico digital, valores de tensão são descritos como conjuntos de palavras digitais, podendo assim ser processados, ou armazenados, em microcontroladores.

O princípio de conversão usa circuitos comparadores, que podem ser construídos com o uso de amplificadores operacionais, para a geração da palavra digital referente a um valor de tensão analógico aplicado na entrada do conversor. Durante a comparação, caso o valor aplicado na entrada do comparador supere o valor de referência, seu sinal de saída representará nível lógico alto, incrementando o valor da palavra digital, caso o valor não supere a comparação retorna nível lógico baixo. Um exemplo de comparador com amplificador operacional pode ser visto na *Figura 1*.

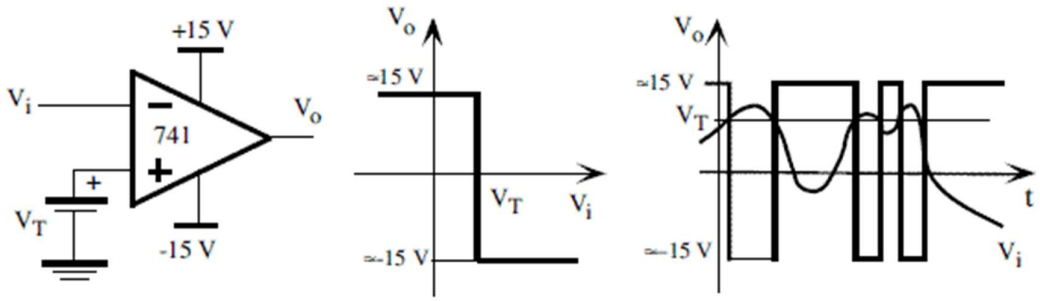


Figura 1 - Exemplo de comparador

O número de comparadores empregados na construção do conversor indica a resolução em bits da palavra digital gerada. Assim, quanto maior o número de comparadores, mais sensível é o conversor.

2.1.2 Medição de tensão alternada

Um método para a medição de tensão por meio de um instrumento que utiliza de um circuito conversor AD pode ser feito a partir do condicionamento do valor de tensão a ser medida para valores compatíveis com os limites de leitura do conversor, por meio de um divisor resistivo que realiza a atenuação do sinal de tensão a ser medido. Deve-se levar em consideração a escolha de resistores de elevado valor, sendo idealmente infinita, para que o dreno de corrente por parte deste conjunto seja baixo, de forma que pouco afete o circuito no qual é realizada a medida.

Na *Figura 2* é mostrado um circuito atenuador em que o valor de tensão que será convertido é dado pela queda de tensão sobre um resistor de referência.

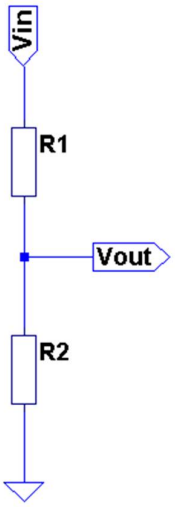


Figura 2 - Circuito atenuador para medição de tensão

A medição de tensão alternada pode ser feita da mesma forma, levando-se em consideração a escolha do ADC e a adequação do sinal de tensão alternada. Uma alternativa para ADC de operação com fonte de alimentação simples é realizar a retificação do sinal de tensão alternada antes da sua atenuação, como mostra a *Figura 3*.

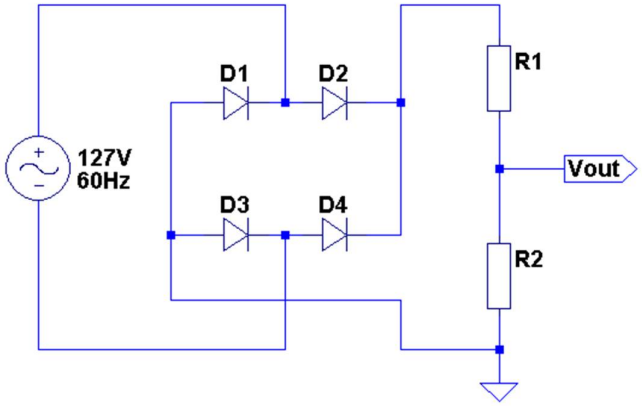


Figura 3 - Circuito condicionador de sinal de tensão alternada

2.1.3 Medição de corrente

Dentre as técnicas para medição de corrente elétrica por meio de um instrumento digital, é por meio do condicionamento da medida de interesse, para um valor de tensão, para que seja feita a conversão por um ADC. Isto pode ser feito com o uso de um resistor de referência, chamado de resistor *shunt*, por onde o sinal de corrente passará, gerando uma queda de tensão sobre este elemento, proporcional ao seu valor, que será

lida por um conversor AD. De forma a não interferir no circuito de interesse, este resistor de referência deve possuir o menor valor possível, sendo idealmente igual à zero. Na *Figura 4* é representado um circuito para medição de corrente por meio da tensão sobre um resistor *shunt*.

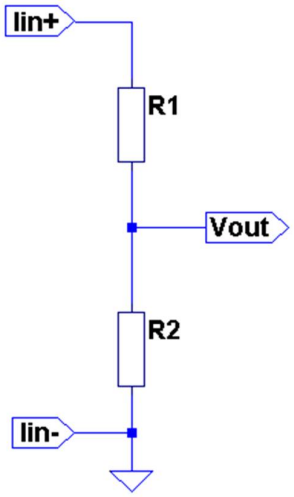


Figura 4 - Circuito condicionador de sinal de corrente

Outro método para determinação de corrente elétrica que circula uma carga é baseada no princípio do efeito Hall, descoberto por Edwin H. Hall em 1879. De acordo com este princípio, um fluxo de corrente em um condutor faz surgir uma diferença de potencial transversal a corrente, e um campo magnético perpendicular a esta. A partir do valor de tensão é possível estimar o valor de corrente elétrica, dado o campo. Uma representação do efeito Hall pode ser vista na *Figura 5*.

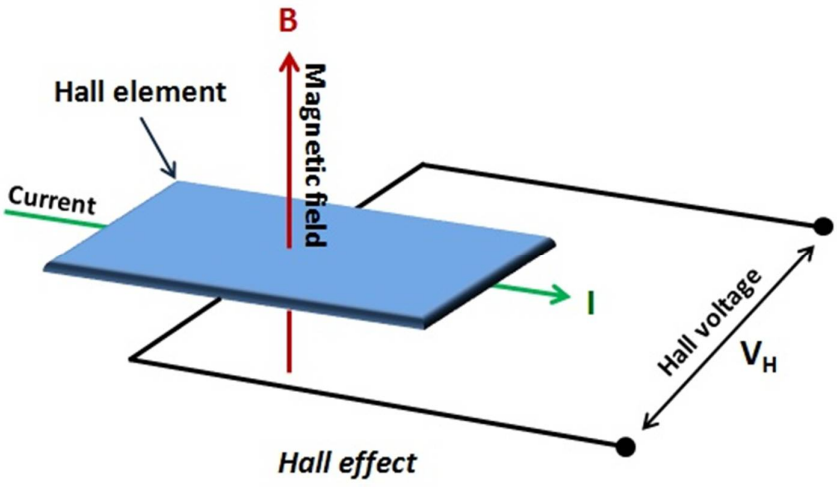


Figura 5 - Representação do efeito Hall. Disponível em: <http://www.unicrom.com/Tut_comparadores_voltaje.asp>

2.1.4 Medição de Potência

A potência elétrica é uma grandeza definida como sendo o trabalho realizado pela corrente elétrica, ou então, a energia elétrica, em um dado período de tempo. É definida pelo produto da tensão e corrente elétrica que atuam em uma carga, descrita pela equação $P = V.I$, sendo V o valor de tensão em volts, e I o valor de corrente em ampères.

Para sistemas de corrente alternada (AC) senoidal, a potência elétrica, chamada de ativa, é calculada pelo produto dos valores médios quadráticos (RMS) de tensão e corrente que atuam sobre a carga, além do cosseno do ângulo de defasagem entre estes sinais, chamado de fator de potência. Quando o cálculo de potência não leva em consideração o fator de potência, obtém-se o valor de potência aparente, valor este utilizado no dimensionamento de cabeamentos e sistemas de proteção em instalações elétricas. Existe também um terceiro tipo de potência, denominada potência reativa, consumida por reatâncias, indutivas ou capacitivas, que pode ser descrita como a potência na qual não se realiza trabalho útil.

Para cálculo e determinação dos três valores de potência, basta possuir os valores de tensão e corrente eficaz, além do fator de potência. Uma representação chamada triângulo de potências, ilustrada na *Figura 6*, relaciona os três tipos de potência e pode ser utilizada para obtenção de seus valores, dada uma delas e o fator de potência.

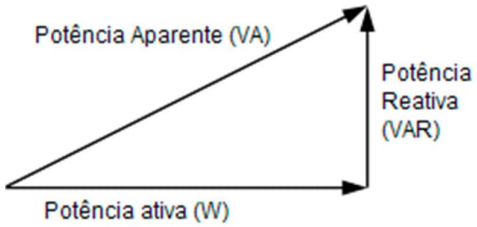


Figura 6 - Triângulo de potências. Disponível em <http://pt.wikipedia.org/wiki/Potência_elétrica>

2.2 Microcontroladores

Microcontroladores são a junção de processadores com outros periféricos, tornando este tipo de componente extremamente versátil e robusto, não necessitando de muitos outros componentes adicionais para seu funcionamento.

Os principais periféricos presentes nos microcontroladores atuais são memória RAM (*Random Access Memory*), memória EEPROM (*Electrically-Erasable Programmable Read-Only Memory*), além de periféricos responsáveis por comunicação, como UART (*Universal asynchronous receiver/transmitter*) e I2C (*Inter Integrated Circuit*).

Da mesma forma que os processadores, os microcontroladores executam funções programadas por um usuário, geralmente armazenadas em uma memória não volátil como a EEPROM. Estas funções podem ser escritas em diversos tipos de linguagem dependendo da família de microcontroladores utilizada, assim como seu compilador.

O uso de linguagens de programação padrão permite de forma simplificada a troca de microcontroladores, dada à necessidade de um projeto e seu desenvolvimento, caso exista a necessidade de maior quantidade de memória ou velocidade de processamento, assim como atualização em para componentes mais novos e baratos.

Para a realização do projeto proposto foi utilizado o microcontrolador ATmega32U4 [4], da *Atmel Corporation*, compatível com a plataforma *Arduino*, sendo o microcontrolador presente na placa de referência *Arduino Leonardo*.

2.2.1 Plataforma *Arduino*

Arduino [5] é uma plataforma para microcontroladores de *hardware* e *software* de arquitetura aberta criada com o intuito de ser uma opção barata e simples para o desenvolvimento de protótipos rápidos, além de facilitador para um primeiro contato com eletrônica e programação para pessoas que nunca tiveram contato com estas áreas de conhecimento. Sua interface de desenvolvimento (IDE) é desenvolvida em linguagem *Java* está disponível para os sistemas operacionais *Windows, Linux* e *Mac OS* e permite a programação por meio das linguagens C e C++ para microcontroladores *Atmel*, além da linguagem *Wiring*.

O *hardware* necessário para o uso da plataforma *Arduino* pode ser adquirido na forma de placas de referência montadas, ou então podem ser montadas versões compatíveis, uma vez que as licenças de uso da plataforma são abertas e permitem a criação e adaptação por parte dos usuários.

A placa utilizada no desenvolvimento do projeto consiste em uma versão da placa de referência *Arduino Leonardo* criada para possuir dimensões pequenas e que pudesse ser utilizada em matrizes de contato, além da capacidade de ser incorporada a placas de circuito impressos para circuitos e projetos maiores com facilidade.

A *Figura 7* apresenta a placa utilizada.

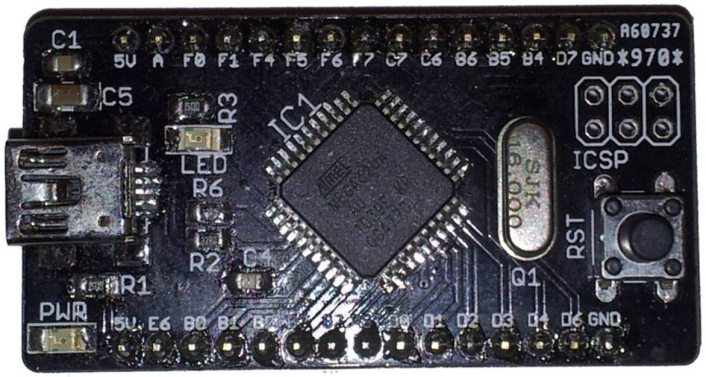


Figura 7 - Placa com microcontrolador ATmega32U4

Abaixo são apresentadas as principais características do microcontrolador *Atmel* ATmega32U4 utilizado no projeto:

- 32 KBytes de memória *Flash*;
- 1 KByte de memória EEPROM;
- 2,5 KBytes de memória SRAM (*Static Random-Access Memory*);
- 1 Timer/Counter de 8-bit;
- 2 Timer/Counter de 16-bit;
- 4 canais de PWM (*Pulse-Width Modulation*) de 8-bit;
- 4 canais de PWM com resolução programável entre 2 e 16 Bits
- 6 canais de PWM para operação em alta velocidade, com resolução programável entre 2 e 11 Bits
- 26 pinos de entrada e saída
- 1 USB (*Universal Serial Bus*) *Transceiver*
- Máxima frequência de operação de 16 MHz
- ADC de 12 canais e 10-bit com recursos de canais diferenciais e ganho programável;

- Serial USART (*Universal Synchronous/Asynchronous Receiver/Transmitter*)
- Tensão de operação: 2,7 V a 5,5 V;
- Temperatura de operação: -40º C a 85º C.

2.2.2 *Google Android*

O *Google Android*, versão atual 5.0, é um sistema operacional para dispositivos móveis, como *smartphones* e *tablets*, desenvolvido pela *Google Inc*. Atualmente este sistema está presente em mais de um bilhão de aparelhos em todo o mundo e possui como grande característica ser baseado no sistema operacional *Linux*, além de ser um sistema aberto. Desta forma, o desenvolvimento de aplicações para este sistema conta com uma grande comunidade de usuários e ferramentas, o que auxilia na criação de novas aplicações.

A criação de aplicações faz uso da plataforma de desenvolvimento de *software Android SDK* [6], que contém as bibliotecas e APIs (*Application Programming Interface*) necessárias para a criação e compilação de aplicações para o sistema. Esta ferramenta pode ser encontrada no *website* oficial da plataforma para desenvolvedores *Android*, e seu *download* é gratuito.

Para a construção do aplicativo para *Google Android* foi utilizado um *framework* chamado *PhoneGap* [7], que além de facilitar a criação de aplicações, permite o acesso aos recursos de *hardware* presentes em dispositivos móveis, permitindo a comunicação entre dispositivos, por exemplo, por meio do protocolo *Bluetooth*.

2.2.2.1 *PhoneGap*

O *PhoneGap* é um *framework* baseado na plataforma *Apache Cordova* que permite a criação de aplicações para sistemas móveis, como *Google Android*, *Windows Phone* e *iOS*, com o uso de linguagens de tecnologias web, como *HTML5*, *CSS* e *Javascript*, permitindo que o usuário crie aplicações sem fazer o uso das linguagens nativas do sistema operacional para o qual está desenvolvendo, além de permitir facilmente a criação de *software* para múltiplos sistemas.

O acesso, por parte do *PhoneGap*, a recursos de *hardware* do dispositivo que executará o programa permite a criação de aplicações que façam uso de acesso a arquivos e dados presentes em sua memória, informações de GPS (*Global Positioning System*), calendário, conectividade *WiFi*, além de acesso a rede de dados telefônica.

Outra característica interessante do *PhoneGap* é a possibilidade do uso de *plug-ins* que permitem o acesso a outros recursos presentes em alguns aparelhos, mas que não são suportados nativamente pelo *framework*, como o suporte a comunicação com outros dispositivos por meio do protocolo *Bluetooth*, utilizada neste projeto.

2.2.2.2 *Heroku*

Heroku [8] é uma plataforma online para execução de aplicações via *web*, que podem ser escritas nas linguagens *Ruby*, *Node.js*, *Python* [9], *Java* e PHP. Dentre os recursos presentes nesta plataforma está o *Postgres*, uma base de dados SQL que permite a manipulação simples de informação permitindo a fácil integração destas com a aplicação desenvolvida.

2.2.2.3 *Python*

Python é uma linguagem de programação de alto-nível criada com a ênfase em possuir facilidade de entendimento, além de conceitos de sintaxe que permitam a construção de aplicações com um menor número de linhas do que seriam necessárias em linguagens como C++ ou *Java*. Esta linguagem possui ainda suporte a múltiplos paradigmas de programação, como orientação a objeto, programação procedural, imperativa, ou então funcional.

A linguagem *Python*, também pode ser utilizada para a construção de aplicações web, cujo desenvolvimento pode ser facilitado com o uso de *frameworks*, como o *Django* [10], um *framework* gratuito e aberto que permite a criação de aplicações *web* de forma rápida, e com alto desempenho.

3. Descrição do projeto e Implementação

Neste capítulo são apresentados os passos para a implementação e desenvolvimento do projeto, assim como seu detalhamento, além dos materiais e métodos empregados.

3.1 Descrição do Projeto

O projeto desenvolvido é composto de um dispositivo de medição das grandezas elétricas de tensão e corrente alternada, além do fator de potência, com suporte a comunicação *Bluetooth* para o envido destas medidas a um dispositivo com suporte a sistema operacional *Google Android*, que fará a exibição em tela dos valores de potência real, além de enviar as informações a um banco de dados, para que sejam exibidos em uma página *web* que poderá ser acessada remotamente por meio de um navegador de internet.

O *hardware* construído é baseado em uma placa com o microcontrolador *Atmel* ATmega32U4, que possui conversor analógico digital de 12 canais e resolução de 10-bit. Assim, foram projetados circuitos condicionadores de sinal que permitissem a leitura de sinais de tensão e corrente alternada para a medição de potência em cargas.

Um relê presente na placa permite que a carga seja desligada remotamente, recurso este ainda não implementado. Também foi utilizado um *display* de OLED (*Organic Light-Emitting Diode*) que exibe as medições em tempo real, para que o usuário possa fazer leituras pontuais com maior facilidade. O módulo de *display* OLED utilizado pode ser visto na *Figura 8* a seguir.

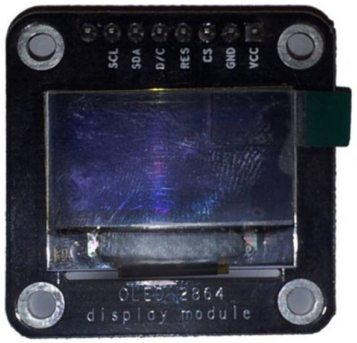


Figura 8 - *Display* **OLED**

Para a criação da aplicação móvel foi escolhido o sistema operacional *Google Android*, por estar presente em uma série de dispositivos, além da disponibilidade de ferramentas de desenvolvimento gratuitas e o suporte a comunicação sem fio por meio do protocolo *Bluetooth*, utilizado no projeto. Como vantagem do uso da comunicação via *Bluetooth* sobre outros protocolos sem fio, pode-se listar o baixo consumo de energia e existência de módulos com suporte a comunicação serial UART, que simplificam o uso em plataformas embarcadas.

O desenvolvimento da aplicação para dispositivos *Google Android* foi feita com o uso do *framework PhoneGap*, que permitiu o uso das linguagens de tecnologia web HTML5, CSS e *javascript* no projeto. A compilação do programa foi feita com o *Android SDK*.

3.2 Materiais Utilizados

Os componentes empregados na montagem do *hardware* do projeto são listados na *Tabela 1*, apresentada a seguir:

Tabela - Componentes utilizados

Componente	Quantidade
Porta-fusível	1
Fusível 1ª	1
Diodo 1N4007	5
Transistor BC548B	1
Resitor 10 kΩ	1
Resistor 100 kΩ	1
Resistor 2,2 kΩ	1
Relê 15A/120VAC - 6VDC	1
Terminal	2
Módulo Bluetooth Serial	1
Display OLED	1
ATmega32U4	1
Conector mini USB	1
Resistor 22 Ω	2
Capacitor 18pF	2
Cristal 16 MHz	1
Capacitor 1 µF	1
Resistor 1 kΩ	1
Capacitor 0.1 µF	1
Capacitor 10 µF	1
LED	2
Chave Tactil	1
Resistor	2
Barra de pinos 15 vias	2

A criação e compilação do aplicativo *Android* desenvolvido foi feita em um computador executando a distribuição *Linux Mint* [11] *Cinnamon* 32-Bit. Os testes foram realizados em um *smartphone Samsung Galaxy SII* com sistema *Google Android 4.4*.

O programa embarcado para microcontrolador ATmega32U4 foi desenvolvido com o uso da *Arduino IDE 1.0.6*, assim como a programação do dispositivo.

3.3 Desenvolvimento do *Hardware*

Para a realização das medições foi projetada uma placa de condicionamento sinais, ligada a placa microcontrolada com *software* embarcado responsável pelo cálculo das medições envio via protocolo *Bluetooth.* A comunicação *Bluetooth* é realiza

com o uso de um módulo conectado ao periférico serial UART presente no microcontrolador, sendo sua utilização bastante simples.

Outro periférico interno ao microcontrolador ATmega32U4 bastante utilizado no projeto é o ADC de 12 canais e 10-bit de resolução, sendo este responsável por converter os valores de tensão analógica, dada a tensão de referência V_{REF}, igual a tensão de alimentação do microcontrolador nesta aplicação, em valor binário de 10-bit, ou 1023 em sua representação decimal. Assim, a cada incremento no sinal lido pelo conversor AD na razão de $V_{ref}/2^{10}$, ocorre o incremento em uma unidade no valor binário que o representa. Para a aplicação realizada com um valor de alimentação de 5 volts, obtém-se uma tensão de passo igual a 4,883 mV, ou seja, cada incremento no valor do sinal de entrada nesta grandeza gera a adição de uma unidade no valor binário da palavra digital que o representa.

Um recurso presente no microcontrolador utilizado é a existência de uma tensão de referência interna de 1,1 volts , tensão esta que pode ser usada de forma a permitir que leituras mais precisas sejam realizadas, como será explicado posteriormente.

A tensão de alimentação escolhida para operação da placa projetada foi de 5 volts, já que esta é a tensão de operação do circuito integrado ACS712-30A [12], fabricado pela *Allegro MicroSystems*, responsável pela medição de corrente. Para alimentação do módulo *Bluetooth* e *display* de OLED, que operam em 3,3 volts, foi utilizado um regulador de tensão LM1117-3.3 que fornece a tensão 3,3 volts a partir dos 5 volts que alimentam o microcontrolador e o sensor de corrente.

O módulo *Bluetooth Serial* utilizado na comunicação do microcontrolador com os dispositivos *Android* é baseado no circuito BC417, fornecido pela *Cambridge Silicon Radio*, e é mostrado na *Figura 9*.

Figura 9 - Módulo *Bluetooth*

As características desse sistema são as seguintes:

- Comunicação USART - Serial RS232 nível TTL;
- Frequência de operação 2,4 GHz Banda ISM (*Industrial, scientific and medical*);
- Modulação GFSK (*Gaussian Frequency Shift Keying*);
- Potencia de Transmissão: Classe 2, < 2,5 mW (4 dBm);
- Alcance de até 10 metros;
- Sensibilidade < -84 dBm à 0.1% BER (*Bit Error Rate*);
- Velocidade: Assíncrona: 2,1 Mbps (Max) / 160 kbps, Síncrona: 1 Mbps / 1 Mbps;
- Tensão de alimentação: 3,6 - 6VDC 50 mA;
- Temperatura de operação: -20°C ~+75° C.

No *Apêndice A* é apresentado o esquemático do circuito completo.

Serão agora detalhados de forma separada os circuitos condicionadores de sinal para medição de tensão e corrente alternada.

3.3.1 Voltímetro

A medição de tensão alternada é realizada por meio da leitura da queda de tensão sobre um resistor de referência constituinte de um circuito atenuador de tensão construído com dois resistores, após uma etapa de retificação de onda completa do sinal

de tensão alternada, feito com uma ponte de diodos. O conjunto é mostrado na *Figura 10* a seguir.

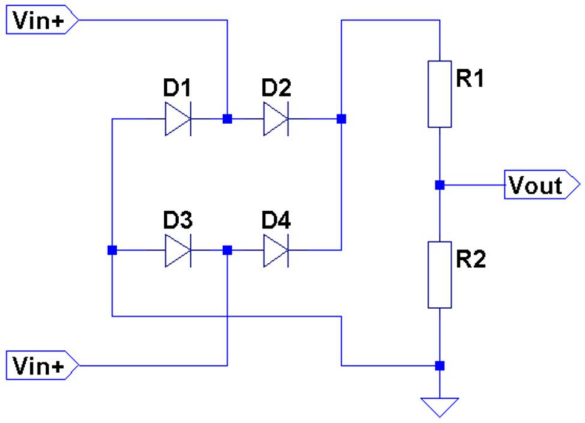


Figura 10 - Circuito condicionador de tensão alternada

3.3.2 Amperímetro

Na obtenção das medidas de corrente alternada que percorrem uma carga foi empregado o circuito ACS712, fabricado pela *Allegro MicroSystems*. Este circuito é uma solução barata, de pequenas dimensões, o que permite a construção de circuitos bastante compactos, além de fácil implementação. Seu princípio de funcionamento é o efeito Hall, descrito anteriormente, e está disponível em três versões para medição de corrente na faixa de mais ou menos cinco, 20, ou 30 Ampères, dependendo da necessidade do usuário. Com o aumento da faixa de corrente máxima a sensibilidade do componente diminui proporcionalmente, sendo respectivamente igual a 185mV/A, 100mV/A e 66mV/A.

Dentre as características deste circuito integrado, pode-se listar:

- Tempo de resposta a uma entrada de corrente igual a 5µs
- Largura de banda de 80kHz;
- Erro total de saída de 1,5% a temperatura de 25ºC;
- 1,2mΩ de resistência do condutor interno;
- Isolação mínima entre os pinos de 2,1kVRMS;
- Alimentação simples de 5V;
- Saída proporcional a corrente de entrada AC ou DC.

De forma a suportar uma maior faixa de aplicações foi escolhido o ACS712-30, cuja resposta de tensão, dada a corrente de entrada, pode ser vista na *Figura 11*.

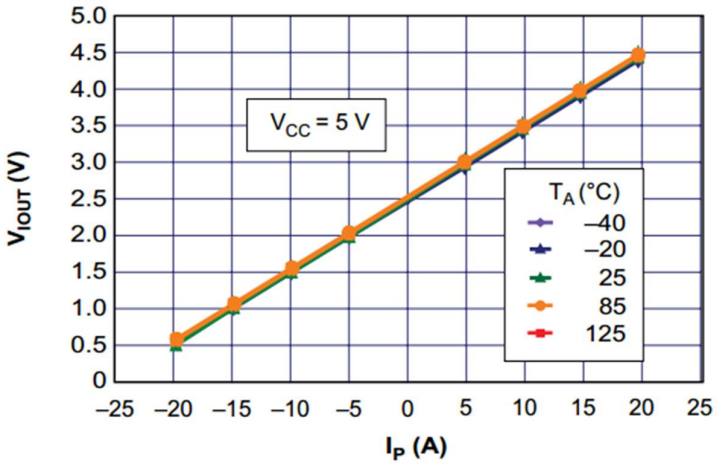


Figura 11 - Curva de tensão de saída do circuito ACS712-30ª. Disponível em <https://www.mpja.com/download/acs712.pdf>

Para corrente igual a zero a tensão de saída do sensor é igual a $^{V_{cc}}/_{2}$, que para a tensão de alimentação de 5V é igual a 2,5V. Desta forma, para valores de corrente maiores ou menores que zero, a tensão de saída varia linearmente em torno deste ponto. Portanto, para o cálculo da corrente, basta obter o valor absoluto da subtração do valor medido instantaneamente pelo valor médio da tensão de alimentação, e dividir pelo valor de escala, que é de 66mV/A para o ACS712-30.

3.3.3 Construção da Placa para medições

A placa que contém o microcontrolador ATmega32U4 e a do circuito condicionador de sinal foram desenhadas com o uso do *software CADSoft Eagle PCB Design* [13]. Os desenhos obtidos foram utilizados para a confecção das placas.

A placa para medições projetada contém, além dos circuitos condicionadores de sinal de tensão e corrente, um fusível para proteção da carga, bornes para ligação de fios de alimentação e saída e um relê para atuação na carga. A vista superior da placa montada com os componentes pode ser vista na *Figura 12*, e a vista inferior na *Figura 13*.

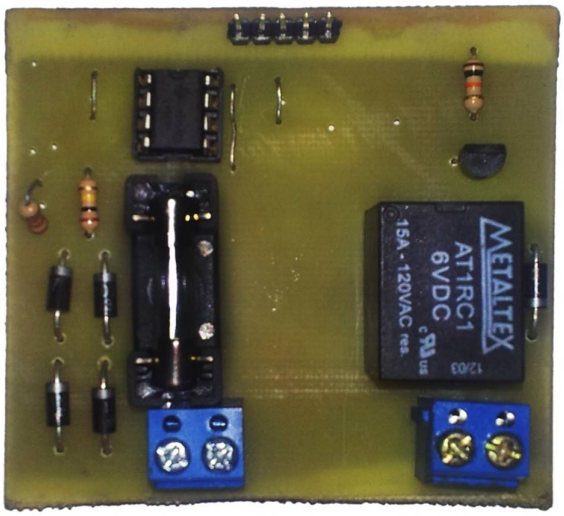


Figura 12 - Vista superior da placa desenvolvida

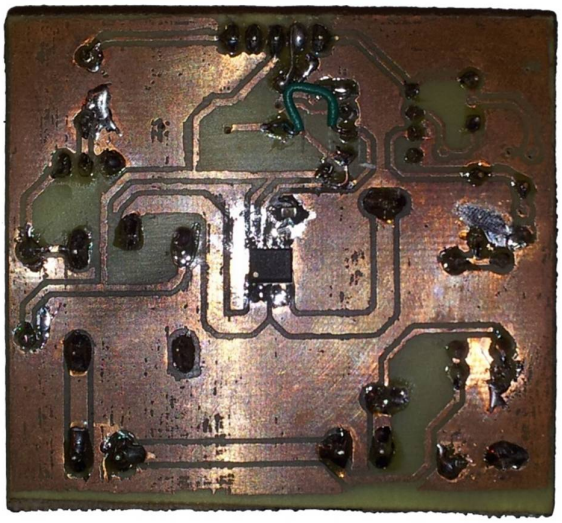


Figura 13 - Vista inferior da placa desenvolvida

3.4 *Software*

3.4.1 Preparação das ferramentas de trabalho para o desenvolvimento do *software*

O desenvolvimento do projeto foi realizado com o uso de um computador com sistema operacional *Linux*, com a distrubuição *Linux Mint*, baseada no sistema *Debian*. Isto foi feito uma vez que este sistema traz como facilidade o suporte às ferramentas que

permitem a criação de aplicações móveis para sistema *Google Android* com documentação vasta, além de uma série de exemplos.

Para a instalação do *Android SDK*, responsável pela compilação do programa que será executado nos dispositivos que rodam o sistema operacional *Android*, é necessário previamente instalar o *Oracle Java SDK*. Para isto, após a obtenção dos arquivos de instalação, é feita a execução dos comandos no terminal de comandos do *Linux*, listados na *Figura 14*.

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

Figura 14 - Lista de comandos para instalação das ferramentas

Este último comando listado fará com que seja aberto o gerenciador de pacotes do *Android SDK*, *Figura* 15, ferramenta em que será feita a seleção e de ferramentas e pacotes para a versão do *Android* para a qual será desenvolvida a aplicação.

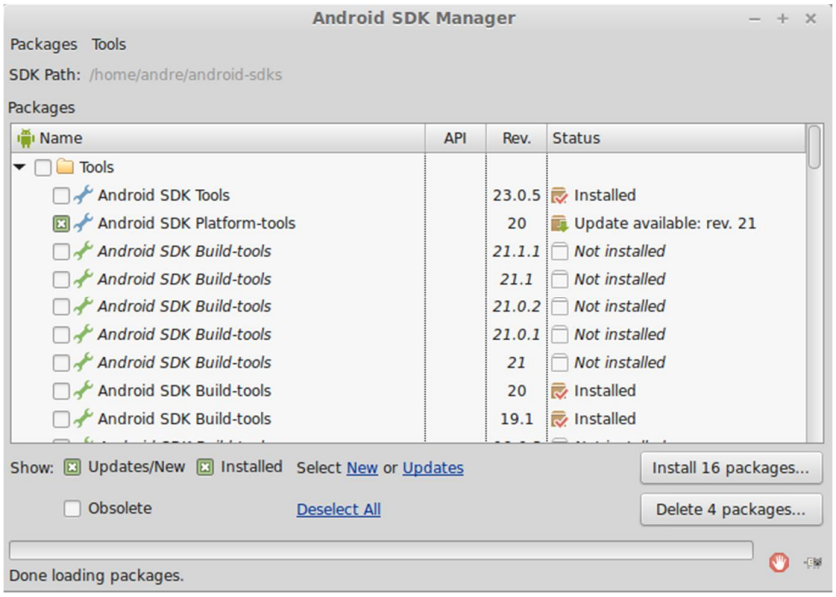


Figura 15 - *Android SDK*

As outras ferramentas empregadas na construção do aplicativo para *Android* são instaladas com o gerenciador de pacotes NPM (*Node Package Module*) [14], que já é incorporado ao sistema quando instalado o *Node.js* [15]. É por meio do gerenciador NPM que são instaladas as ferramentas *Apache-cordova* e *PhoneGap*, como mostra a *Figura 16*.

```
#  npm install –g cordova
#  npm install –g phonegap
```

Figura 16 - Comandos instalação *PhoneGap*

Após isto já é possível criar aplicações com o *PhoneGap* e compilar para as plataformas de interesse. Os comandos necessários para isto são mostrados a seguir, na *Figura 17*.

```
#  phonegap create wattimetro
#  cd wattímetro
#  cordova platform add android
#  cordova plugin add
```

Figura 17 - Comandos para criação de aplicação com o uso do *PhoneGap*

A *Arduino IDE*, utilizada para desenvolvimento do programa embarcado no microcontrolador *Atmel* ATmega32U4, pode ser baixada no site oficial da plataforma. Já a biblioteca para uso do *display* OLED SSD1306 [16] pode ser encontrada em seu

repositório no endereço https://github.com/adafruit/Adafruit_SSD1306 e deverá ser adicionada juntamente as demais bibliotecas para *Arduino* localizada na pasta que contém os arquivos de execução da IDE.

3.4.2 Programa embarcado

O código para o microcontrolador *Atmel* ATmega32U4 foi desenvolvido por meio da IDE *Arduino*, e o código consiste na constante leitura dos conversores analógico-digitais responsáveis pela medição de tensão e corrente alternada que circula a carga de interesse, segundo os circuitos condicionadores de sinal.

Após a leitura destes valores, são então calculados os dos valores de tensão e corrente baseado nas contas, apresentadas anteriormente, para que sejam obtidos os valores que atuam na carga. De forma a obter medidas mais precisas foi empregado um algoritmo, *Figura 18*, que utiliza de um circuito de referência interno ao microcontrolador ATmega32U4, cujo valor constante de 1,1V é utilizado em uma rotina para se obter o valor de tensão de referência do conversor AD de maneira mais precisa.

```
long readVcc() {
  // Lê referência interna de 1.1V e compara com AVcc
  // Configura a referência para Vcc e a mede a referencia
interna de 1.1V
  ADMUX = _BV(REFS0) | _BV(MUX4) | _BV(MUX3) | _BV(MUX2) |
_BV(MUX1);

  delay(2); // Aguarda até Vref estabilizar
  ADCSRA |= _BV(ADSC); // Inicia conversão
  while (bit_is_set(ADCSRA,ADSC)); // medição

  uint8_t low  = ADCL; // primeiramente lê bits menos
significativos - isto trava os bits mais significativos
  uint8_t high = ADCH; // destrava todos bits

  long result = (high<<8) | low;

  result = 1125300L / result; // Calcula Vcc em mV 1125300
= 1.1*1023*1000
  return result; // Retorna Vcc em milivolts
}
```

Figura 18 - Algoritmo para calculo da tensão AREF

Para o cálculo dos valores médios quadráticos (RMS) de tensão e corrente, para a obtenção dos valores de potência e fator de potência, assim como potência aparente, foram utilizadas as formulações disponíveis no *website OpenEnergyMonitor* [17] <http://openenergymonitor.org/emon/buildingblocks/ac-power-arduino-maths>, que contém informações para a construção de dispositivos para monitoramento de consumo de energia.

Inicialmente, para cálculo da potência real, devem ser medidos os valores instantâneos de tensão e corrente que circulam a carga de interesse, que multiplicados resultam em um valor de potência instantânea. Ao realizar este cálculo por um determinado número de amostras, e calculada a média do valor obtido, o valor resultante representa o valor de potência real consumido pela carga. Na *Figura 19* pode ser visto o algoritmo para cálculo de potência real.

```
for (n=0; n<numero_de_amostras; n++)
{
  potencia_instantane = tensao_instantanea *
corrente_instantanea;
  soma_potencia_instantanea += potencia_instantanea;
}
potencia_real = soma_potencia_instantanea/numero_de_amostras;
```

Figura 19 - Cálculo potência real

Para o cálculo da potência aparente são necessários os valores médios quadráticos de tensão e corrente, que quando multiplicados, representam o valor de potência aparente. Um simples algoritmo para cálculo do valor RMS de tensão alternada pode ser visto na *Figura 20*.

```
for (n=0; n<numero_de_amostras; n++)
{
  tensao_quadratica = tensao_instantanea * tensao_instantanea;
  soma_tensao_quadratica += tensao_quadratica;
}
tensao_media_quadratica = soma_tensao_quadratica/numero_de_amostras;
valor_medio_quadratico_tensao = sqrt(tensao_media_quadratica);
```

Figura 20 - Cálculo valor de tensão RMS

De forma análoga pode ser calculado o valor RMS de corrente alternada, como mostra a *Figura 21*.

```
for (n=0; n<numero_de_amostras; n++)
{
  corrente_quadratica = corrente_instantanea * corrente_instantanea;
  soma_corrente_quadratica += corrente_quadratica;
}
corrente_media_quadratica = soma_corrente_quadratica/numero_de_amostras;
valor_medio_quadratico_corrente = sqrt(corrente_media_quadratica);
```

Figura 21 - Cálculo valor de corrente RMS

Obtidos os valores de tensão e corrente eficazes, basta realizar a sua multiplicação para que se obtenha o valor de potência aparente, *Figura 22*.

```
potencia_aparente = valor_medio_quadratico_tensao *
valor_medio_quadratico_corrente;
```

Figura 22 - Cálculo valor de potência aparente

Já o valor de fator de potência é calculado a partir da divisão do valor de potência real pela potência aparente, *Figura 23*.

```
fator_potencia = potencia_real / potencia_aparente;
```

Figura 23 - Cálculo do fator de potência

Após cálculo dos valores medidos, estes são exibidos em um *display* matricial de OLED de 128 colunas por 64 linhas, com o uso da biblioteca *Adafruit SSD1306*, disponibilizada gratuitamente. Estes valores também são enviados via comunicação serial UART para o módulo *Bluetooth* conectado ao microcontrolador, responsável pela comunicação com os dispositivos *Android*. A *Figura 24* exibe o funcionamento do *display* de OLED utilizado.

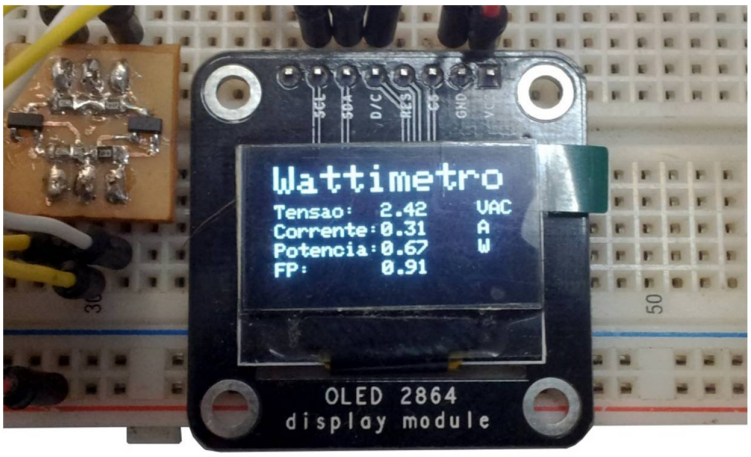


Figura 24 - *Display* OLED em funcionamento

O código completo do programa embarcado está presente no *Apêndice B*

3.4.3 Programa para *Android*

Dentre os recursos que a aplicação para dispositivos móveis *Android* necessitaria possuir, podem ser listados o suporte a comunicação via *Bluetooth*, para recebimento dos dados obtidos pelo *hardware* construído, e acesso a internet, para envio das medições para um banco de dados, para posterior acesso. Com o uso da plataforma *PhoneGap* é possível criar aplicações de forma simples e que permitam o acesso aos recursos necessários para esta aplicação.

O acesso a internet, via conexão *WiFi*, ou via dados por meio de rede de telefonia móvel, é suportado nativamente pelo *PhoneGap* sem que o usuário precise alterar ou adicionar arquivos a estrutura existente da plataforma. Já o acesso a comunicação *Bluetooth* para comunicação com o módulo serial para plataformas embarcadas não é suportado nativamente pelo *PhoneGap*, no entanto um *plugin* chamado *Bluetooth Serial* [18], disponível no site da ferramenta, adiciona este recurso para uso na aplicação.

Com base em um dos exemplos presente no *plugin Bluetooth Serial* foi desenvolvida a aplicação com o uso das linguagens de tecnologia *web HTML*, *CSS* e *javascript*. Assim, após recebimento dos valores de tensão, corrente e fator de potência, medidos pelo instrumento desenvolvido, é então feito o cálculo da potência real

consumida pela carga, valor que é exibido na tela do aplicativo, assim como um gráfico que mostra as medidas de forma contínua ao longo do tempo, conforme as novas leituras são recebidas. Para a exibição em gráfico, foi utilizada uma biblioteca chamada *Flot* [19], que permite a criação de gráficos em páginas *web*, possibilitando o uso na aplicação com o *framework PhoneGap*.

Quando o aplicativo é iniciado, uma tela para escolha do dispositivo *Bluetooth* é mostrada para o usuário, tanto ativação a conectividade *Bluetooth*, quanto o pareamento com o dispositivo alvo devem ser realizados previamente pelo usuário, para que exista a opção de conexão com o dispositivo projetado.

Pressionando o botão "Conectar" é feita a requisição de conexão com o dispositivo *Bluetooth*. Caso a conexão não ocorra, novamente o usuário tem a opção de selecionar um dispositivo para conexão. As telas mencionadas podem ser observadas na *Figura 25*.

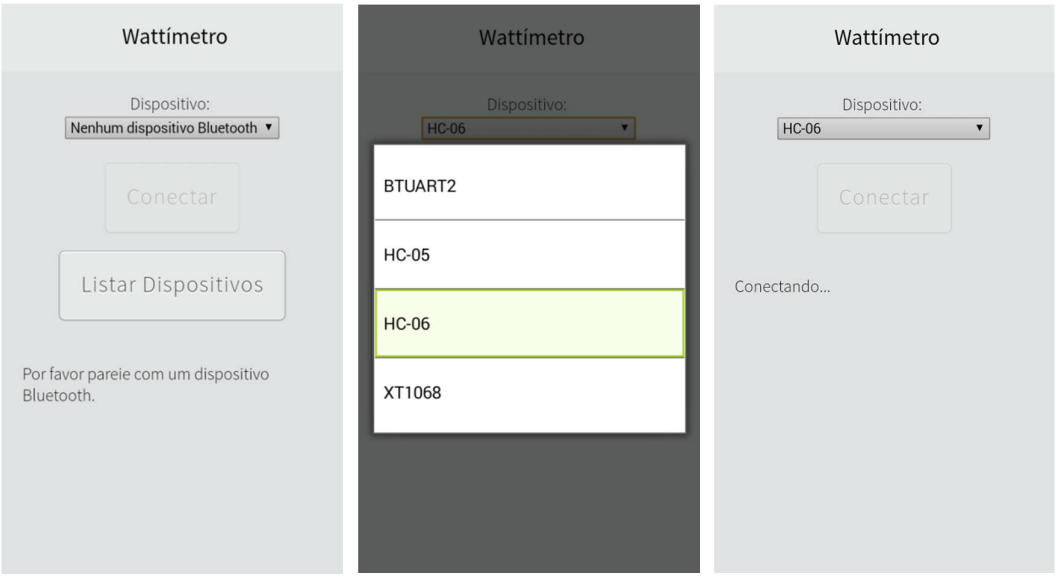


Figura 25 - Tela de seleção de dispositivo

Caso a conexão seja bem sucedida, a aplicação parte para uma segunda tela onde exibe a medição de potência atual, além de um gráfico contendo as leituras ao longo do tempo. Um botão presente abaixo do gráfico com a opção "Desconectar" leva o usuário a tela inicial da aplicação quando pressionado, desfazendo a conexão com o dispositivo *Bluetooth*. Está tela pode ser observada na *Figura 26*.

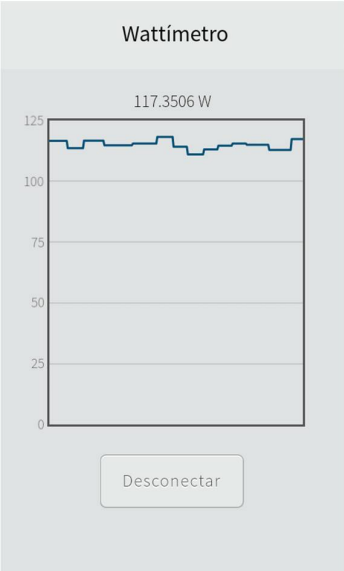


Figura 26 - Tela de exibição de leitura gráfica

Como dito anteriormente foram utilizadas linguagens padrão *web* para o desenvolvimento da aplicação. A seguir serão apresentadas as principais funções utilizadas na construção do programa para *Google Android*.

O arquivo *index.js*, escrito na linguagem *javascript*, contém as principais funções relacionadas a comunicação *Bluetooth* utilizada, a começar pela função *bluetoothSerial.connect*, que é executada quando o usuário pressiona o botão *Conectar*, presente na tela inicial da aplicação. Caso a conexão com o dispositivo *Bluetooth* seja bem sucedida é então chamada a função *app.connect*, responsável por alternar a tela da aplicação para o cálculo e exibição das leituras de potência, em forma de texto e gráfico, assim como envio para banco de dados para posterior acesso, caso o dispositivo *Android* possua conexão a internet ativada no momento da medição. Se eventualmente a conexão com o dispositivo *Bluetooth* não for bem sucedida, é então chamada a função *app.disconnect* responsável por exibir na tela um aviso de que a conexão não foi estabelecida.

A função *onmessage*, executada sempre que uma nova leitura é recebida pelo dispositivo *Android*, possui em sua estrutura uma função para envio das informações via *POST ajax* para o endereço responsável por armazenar as medições na base de dados *Postgres*. Um campo chamado *"device"* contém o nome do dispositivo ao qual o *smartphone* se conecta, sendo atrelado e identificado a um usuário no sistema *web*. Este trecho de código pode ser visto na *Figura 27*.

```
onmessage: function(message) {
  var teste = message;
  var data = teste.split(',');
  tensao = data[0];
  corrente = data[1];
  fp = data[2];
  medicao = tensao * corrente * fp;
  messages.innerHTML = medicao + " W";

  $.ajax({
    type: "POST",
    url: 'http://wattmeter-web.herokuapp.com/publish/',
    data: {
      'device': 'proto',
      'current': corrente,
      'voltage': tensao,
      'fp': fp,
      'created': new Date($.now())
    },
    success: function (data) {
      if (data.success) {
        $('#status').html('SUCCESS');
      } else {
        $('#status').html('ERROR');
      }
    }
  });
}
```

Figura 27 - Envio via *POST* dos valores medidos

O gráfico de medições ao longo do tempo é baseado em um número determinado de medidas passadas previamente armazenadas em um vetor. Assim que novas leituras são recebidas, as leituras já armazenadas são deslocadas e o valor mais antigo é descartado, e então a medida mais atual passa a ocupar a posição referente à medição mais atual neste vetor. Após isto, é então chamada à função responsável pela exibição destes dados no gráfico. Este processo recursivo ocorre de forma que a atualização do gráfico seja feita continuamente ao longo do tempo.

Os códigos referentes à aplicação para dispositivos *Google Android* estão presentes no *Apêndice C*.

3.4.4 Plataforma *web*

Para o desenvolvimento da plataforma *web*, responsável pelo acesso remoto por um usuário as exibições dos valores de medições obtidos, foi criada uma aplicação com o uso da linguagem *Python*, que é executada via plataforma *Heroku*. Para a criação da página também foram empregadas as linguagens *web* HTML, CSS e *javascript*. A aplicação nomeada de “Wattímetro” pode ser acessada por meio do endereço wattmeter-web.herokuapp.com.

Na página inicial o usuário o usuário encontra uma tela de acesso a aplicação, em que necessita realizar *login* para acesso as informações referentes aos seus instrumentos. A tela de *login* pode ser vista na *Figura 28*.

Figura 28 - Tela de *login*

Ao entrar corretamente com seus dados de *login* a próxima página contém opções de acesso aos gráficos das medidas elétricas realizadas por um instrumento, informações dos instrumentos, aqui chamados de "Leitores", além de exportação das informações armazenadas no banco de dados via arquivo CSV (*Comma-separated values*). Este tipo de formato é simples e suportado por diversos *softwares*, como planilhas eletrônicas, ou programas para aplicações matemáticas, como o *Matlab*. Também está presente a opção "Sair", para que o usuário encerre seu acesso a página, permitindo que seja feito o acesso a outra conta.

O cabeçalho com as opções de navegação pode ser visto na *Figura 29*.

Figura 29 - Cabeçalho de navegação

A página inicial após acesso é a referente aos gráficos das medidas. Esta página contém dois gráficos, um de consumo de potência real de uma carga ao longo do tempo, e outro que contém as leituras de tensão e corrente sobrepostas, ao longo do tempo. Uma amostra destes dois gráficos pode ser vista na *Figura 30*, presente a seguir:

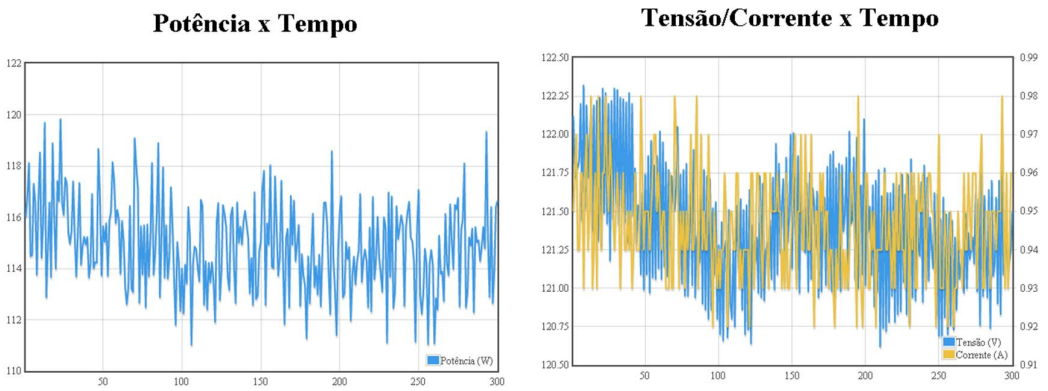


Figura 30 - Gráficos exibidos

Os códigos referentes à aplicação *web* desenvolvida podem ser encontrados no *Apêndice D*.

O fluxograma completo do projeto pode ser visto no *Apêndice E*.

4. Resultados

A seguir são apresentados os resultados obtidos, em relação ao *hardware* desenvolvido, além das aplicações para dispositivos móveis e plataforma *web*.

4.1 *Hardware*

Para a validação do circuito desenvolvido, assim como o programa para dispositivos *Google Android*, e plataforma para acesso das medições via *web*, foram realizados testes de medições em bancada. O teste realizado consistiu na medição de uma carga formada por duas lâmpadas incandescentes de 60 watts 127 V nominais, e registrados os valores obtidos no *display* OLED presente na montagem do circuito de medição. Afim de comparação foi utilizado um multímetro digital de mão comercial para registrar valores cuja leitura e precisão é garantida pelo seu fabricante. Foram computadas um total de 30 medidas, presentes na *Tabela 2*, cujos valores foram utilizados para análise do dispositivo projetado, sendo as medidas representadas por MM (Multímetro) as obtidas por meio de um multímetro, e D (Dispositivo) as obtidas com o dispositivo construído.

Tabela - Valores obtidos para análise

Medição	Tensão MM (V)	Tensão D (V)	Corrente MM (A)	Corrente D (A)	FP
1	121,1	122,42	0,912	0,91	0,72
2	121,1	121,75	0,912	0,94	0,73
3	121,1	121,4	0,912	0,91	0,84
4	121,1	122,13	0,912	0,94	0,74
5	121,1	121,7	0,912	0,92	0,81
6	121,1	121,16	0,912	0,94	0,74
7	121,1	123,15	0,912	0,94	0,78
8	121,1	121,75	0,912	0,94	0,83
9	121,1	122,01	0,912	0,96	0,8
10	121,1	122,14	0,912	0,91	0,81
11	121,1	122,5	0,912	0,92	0,77
12	121,1	123,07	0,912	0,94	0,83
13	121,1	122,86	0,912	0,94	0,77
14	121,1	122,3	0,912	0,92	0,71
15	121,1	121,72	0,912	0,94	0,79
16	121,1	122,4	0,912	0,91	0,82
17	121,1	122,07	0,912	0,93	0,73
18	121,1	121,87	0,912	0,91	0,73
19	121,1	123,08	0,912	0,91	0,82
20	121,1	122,28	0,912	0,91	0,84
21	121,1	122,09	0,912	0,92	0,81
22	121,1	122,69	0,912	0,92	0,74
23	121,1	122	0,912	0,95	0,8
24	121,1	122,33	0,912	0,97	0,75
25	121,1	121,58	0,912	0,93	0,76
26	121,1	121,29	0,912	0,95	0,73
27	121,1	122,37	0,912	0,97	0,72
28	121,1	122,2	0,912	0,91	0,82
29	121,1	122,5	0,912	0,94	0,73
30	121,1	121,27	0,912	0,95	0,8

A partir dos valores de tensão medidos foi traçado um gráfico, *Figura 31*, para comparação das medidas e retirada algumas informações, como erro relativo médio.

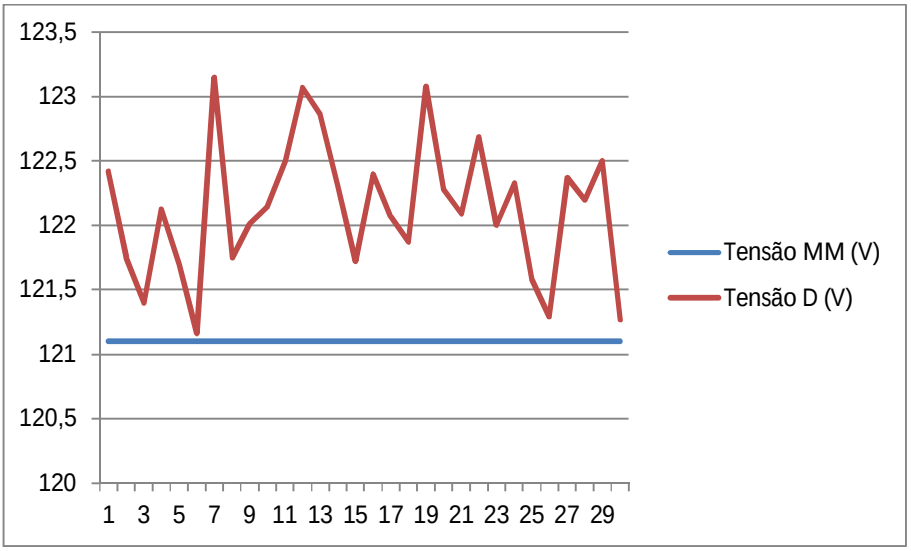


Figura 31 – Gráfico de comparação da medição de tensão

Foi obtido o valor de 0,86% para o erro relativo médio, o que representa um excelente resultado, dado o baixo custo do circuito de condicionamento de sinal projetado.

O mesmo procedimento foi feito com os valores de corrente obtidos, cujo gráfico está presente na *Figura 32*.

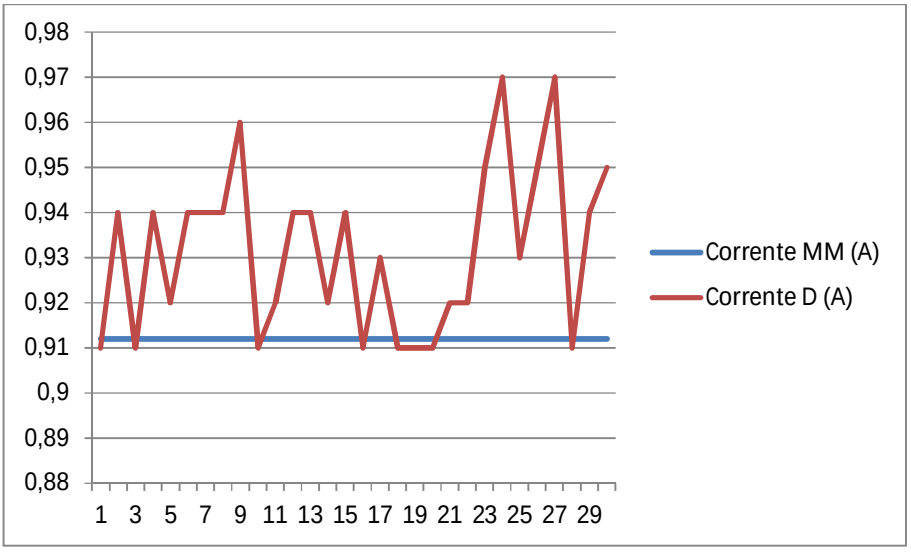


Figura 32 - Gráfico de comparação da medição de corrente

No caso da corrente o erro relativo médio obtido foi de 2,27%, valor maior que o obtido no medidor de tensão, embora continue sendo um resultado positivo dadas as mesmas condições, de um circuito de baixo custo e bastante versátil.

Um fato para o qual se deve atentar é com relação ao fator de potência obtido, *Figura 33*, que além de possuir grande variação, também destoa do esperado, apresentando valores baixos para uma carga idealmente resistiva, ou seja, era esperado um valor próximo de 1.

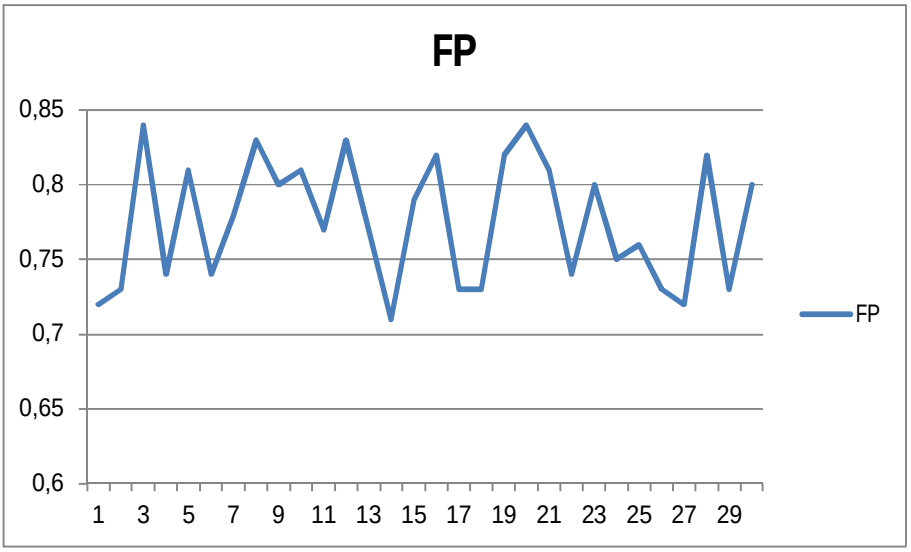


Figura 33 - Gráfico com os valores de fator de potência obtidos

Para explicar isto deve atentar que os valores medidos com o instrumento desenvolvido apresentaram grande variação entre si, diferente do instrumento comercial, que apresentou resultados constantes sem variação na carga. Tanto os valores de tensão, quanto os de corrente se comportaram desta forma devido ao fato de pequenas flutuações na tensão de alimentação do circuito integrado utilizado, a *Allegro* ACS712-30A, além de sua baixa sensibilidade (66mV/A) que não permite que sejam medidos valores baixos de corrente com muita precisão com um conversor analógico digital de baixa resolução. Ruídos presentes no circuito, tal como nos cabos de sinal, além de montagem em matriz de contato da interface para o microcontrolador, também afetam as leituras de interesse.

Como solução dos possíveis problemas observados estão à utilização do circuito integrado ACS712-5A, que possui maior sensibilidade (185mV/A) se comparado com o circuito que permite leituras de até 30 A de pico. A montagem de uma placa que contenha todos os componentes e ligações, assim como a presença de malha de terra,

reduz a presença de ruídos indesejados, reduzindo assim erros de medição. A utilização de fonte de alimentação de maior estabilidade para o circuito de medição também reduziria os erros, já que os circuitos de medição são muito afetados a esta variação.

4.2 *Software*

Tanto o *software* para dispositivos *Google Android*, quanto o desenvolvido para web apresentaram resultados bastante satisfatório na integração dos três sistemas utilizados, sendo eles: *software* embarcado com comunicação sem fio *Bluetooth*, *software* para dispositivos *Google Android*, e por fim, aplicação *web* para acesso e requisição de medições.

5. Conclusão

O principal fundamento de proposta do trabalho realizado era o acesso de forma remota a informações de grandezas elétricas que atuavam em uma carga de interesse, já que este tipo de informação tem sido um alvo cada vez maior para todas as pessoas as quais este tipo de informação pode, mesmo que de forma pouco significativa, afetar. Como dito anteriormente, estudos apontaram que o acesso à informação de padrão de consumo por usuários residenciais resultaram em uma redução de consumo, como simples resultado de um consumo mais consciente, sem que medidas forçadas, como políticas de racionamento, ou aumento de valor de tarifa de consumo fossem tomadas.

Como resultado do trabalho realizado a proposta de *hardware*, embora fundamentalmente correta, na prática não apresentou resultados satisfatórios para medição de fator de potência, a baixa sensibilidade do circuito para medição de corrente impactou tanto a medição desta grandeza. Fontes de ruído, como cabeamento para conexão das interfaces empregadas e matriz de contatos também colaboram para a degradação do sinal a ser lido pelo conversor analógico digital presente no microcontrolador ATmega32U4.

Entretanto, uma vez que o principio do trabalho proposto consistia no acesso remoto as medições realizadas por um instrumento portátil a integração entre sistemas embarcados com plataformas móveis com o intuito de utilização de seus recursos de acesso a internet, sendo ela por meio de rede telefônica, ou *WiFi*, para publicação em um banco de dados associado a uma plataforma para acesso e visualização gráfica, além de exportação dos dados, não apresentou falhas, permitindo assim afirmar a validade da plataforma desenvolvida.

Já com relação ao trabalho desenvolvido como um todo este se mostrou de grande valor para a formação, já que além de possibilitar contato com problemas que diariamente são enfrentados por profissionais que atuam na área de engenharia elétrica, foi uma grande fonte de conhecimento e informações de sistemas, plataformas e recursos computacionais e *web* que não são comumente foco na formação deste tipo de profissional, mas que a cada dia se mostram como um diferencial e permitem a criação de soluções muito interessantes.

Trabalhos Futuros

Como propostas para trabalhos futuros estão o estudo e implementação de circuitos alternativos para a realização de medições de potência elétrica, além da habilitação de atuação na carga, por meio de um relê, para que se desative, ou ative uma carga remotamente, com finalidade de proteção, ou redução de consumo. A extensão para sistemas móveis de diferentes fabricantes, como *Apple iOS*, ou *Microsoft Windows Phone* por meio do desenvolvimento de aplicações dedicadas a estas plataformas também é um ponto interessante, já que permitiria que um maior número de usuários pudesse utilizar o sistema. Incrementos de funções na plataforma *web* também podem ser facilmente realizados, como ajustes nas escalas de exibição dos gráficos e novas informações.

Referências Bibliográficas

[1] Office of Electricity Delivery & Energy Reliability, Centerpoint Energy and U.S Deputy Secretary of Energy Daniel Poneman announce Results of Pilot Project on Home Energy Use. Disponível em: <http://energy.gov/oe/articles/centerpoint-energy-and-us-deputy-secretary-energy-daniel-poneman-announce-results-pilot> Acesso em 29 nov.2014

[2] Computer Wolrd; Residents cut energy bills 11% in smart meter test. Disponível em: <http://www.computerworld.com/article/2510869/it-management/residents-cut-energy-bills-11--in-smart-meter-test.html> Acesso em 29 nov.2014

[3] Android. Disponível em: <http://www.android.com/> Acesso em 29 nov.2014

[4] Datasheet. Atmel AVR ATmega32U4. Disponível em: <http://www.atmel.com/Images/7766s.pdf> Acesso em 29 nov.2014

[5] Arduino. Disponível em: <http://arduino.cc//> Acesso em 29 nov.2014

[6] Android SDK. Disponível em: <http://developer.android.com/sdk/index.html> Acesso em 29 nov.2014

[7] PhoneGap. Disponível em: <http://phonegap.com/> Acesso em 29 nov.2014

[8] Heroku. Disponível em: <https://www.heroku.org/> Acesso em 29 nov.2014

[9] Python. Disponível em: <https://www.python.org/> Acesso em 29 nov.2014

[10] Django. Disponível em: <https://www.djangoproject.com/> Acesso em 29 nov.2014

[11] Linux Mint. Disponível em: <http://www.linuxmint.com/> Acesso em 29 nov.2014

[12] Datasheet. Allegro Microsytems ACS712. Disponível em: <http://www.allegromicro.com/~/media/Files/Datasheets/ACS712-Datasheet.ashx> Acesso em 29 nov.2014

[13] CADSoft Eagle PCB Design. Disponível em: <http://www.cadsoftusa.com/> Acesso em 29 nov.2014

[14] Node Package Modules. Disponível em: <http://www.npmjs.org/> Acesso em 29 nov.2014

[15] Node.js. Disponível em: <http://www.nodejs.org/> Acesso em 29 nov.2014

[16] Adafruit_SSD1306 library. Disponível em: <https://github.com/adafruit/Adafruit_SSD1306> Acesso em 29 nov.2014

[17] OpenEnergyMonitor. Disponível em: <http://openenergymonitor.org/> Acesso em 29 nov.2014

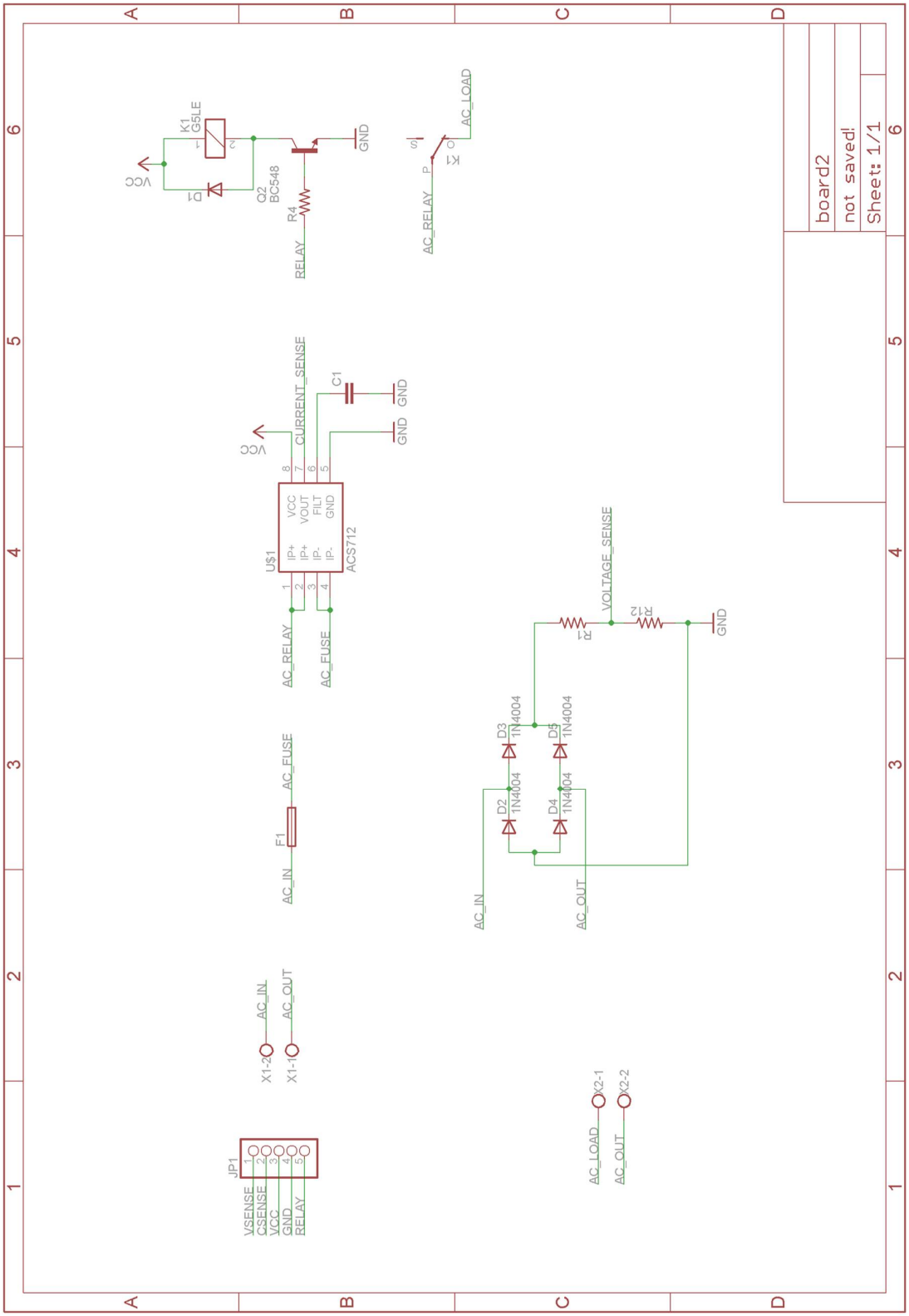
[18] Bluetooth Serial Plugin for Phonegap. Disponível em: <https://github.com/don/BluetoothSerial> Acesso em 29 nov.2014

[19] Flot. Disponível em: <http://www.flotcharts.org/> Acesso em 29 nov.2014

[20] Reis, Priscila dos; Desenvolvimento de disposistivo sem fio para medição de grandezas elétricas para sistemas Google Android. São Carlos, 2014

[21] Fazio Junior, Alírio; Metodologias de medição de energia elétrica reativa (varh) e erros de medição em equipamentos eletrônicos de (varh). Ilha Solteira: [s.n.], 2011

Apêndice A – *Hardware* Desenvolvido



Apêndice B – Código microcontrolador ATmega32U4

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include
<Adafruit_SSD1306.h>

#define OLED_RESET 4
Adafruit_SSD1306
display(OLED_RESET);

#define NUMFLAKES 10
#define XPOS 0
#define YPOS 1
#define DELTAY 2

#if (SSD1306_LCDHEIGHT !=
64)
#error("Height incorrect, please
fix Adafruit_SSD1306.h!");
#endif

#define voltagePin A5
#define currentPin  A0

int number_of_samples = 100;

void setup()  {
 Serial.begin(9600);
 Serial1.begin(9600);

 // by default, we'll generate the
high voltage from the 3.3v line
internally! (neat!)

display.begin(SSD1306_SWIT
CHCAPVCC, 0x3D);  //
initialize with the I2C addr
0x3D (for the 128x64)
 // init done

 //delay(2000);

 // Clear the buffer.
 display.clearDisplay();

 display.setTextSize(2);
 display.setTextColor(WHITE);
 display.setCursor(5,0);
 display.println("Wattimetro");
 display.display();

 display.setTextSize(1);
 display.setTextColor(WHITE);
 display.setCursor(5,20);
 display.println("Tensao: ");
 display.setCursor(110,20);
 display.println("VAC");
 display.display();

 display.setTextSize(1);
 display.setTextColor(WHITE);
 display.setCursor(5,30);
 display.println("Corrente: ");
 display.setCursor(110,30);
 display.println("A");
 display.display();

 display.setTextSize(1);
 display.setTextColor(WHITE);
 display.setCursor(5,40);
 display.println("Potencia: ");
 display.setCursor(110,40);
 display.println("W");
 display.display();

 display.setTextSize(1);
 display.setTextColor(WHITE);
 display.setCursor(5,50);
 display.println("FP: ");
 display.display();

}

void loop() {
 float voltage =
rms_voltage(number_of_sample
s);
 float current =
rms_current(number_of_sample
s);
 float apparent_power = voltage
* current;
 float power =
real_power(number_of_samples
);
 float power_factor = power /
apparent_power;

 Serial.println(String(voltage) +
',' + String(current) + ',' +
String(power_factor));
 Serial1.println(String(voltage)
+ ',' + String(current) + ',' +
String(power_factor));

 display.fillRect(60, 20, 50, 10,
BLACK);
 //display.display();
 display.setTextSize(1);
 display.setTextColor(WHITE);
 display.setCursor(60,20);
 display.println(voltage);
 display.display();

 display.fillRect(60, 30, 50, 10,
BLACK);
 display.setTextSize(1);
 display.setTextColor(WHITE);
 display.setCursor(60,30);
 display.println(current);
 display.display();

 display.fillRect(60, 40, 50, 10,
BLACK);
 display.setTextSize(1);
 display.setTextColor(WHITE);
 display.setCursor(60,40);
 display.println(power);
 display.display();

 display.fillRect(60, 50, 50, 10,
BLACK);
 display.setTextSize(1);
 display.setTextColor(WHITE);
 display.setCursor(60,50);
 display.println(power_factor);
 display.display();

 //delay(500);
}

long readVcc() {
 // Le referencia interna de 1.1V
e compara com AVcc
 // Configura a referencia para
Vcc e a made a referencia
interna de1.1V
 ADMUX = _BV(REFS0) |
_BV(MUX4) | _BV(MUX3) |
_BV(MUX2) | _BV(MUX1);

 delay(2); // Aguarda até Vref
estabilizar
 ADCSRA |= _BV(ADSC); //
Inicia conversao
 while
(bit_is_set(ADCSRA,ADSC));
// meducao

 uint8_t low  = ADCL; //
primeiramente le bits menos
significativos - isto trava os bits
mais significativos
 uint8_t high = ADCH; //
destrava todos bits

 long result = (high<<8) | low;

 result = 1125300L / result; //
Calcula Vcc em mV 1125300 =
1.1*1023*1000
 return result; // Retorna Vcc
em milivolts
}

float read_voltage(int
voltagePin) {
 float vref = readVcc() / 1000;
 float volt =
analogRead(voltagePin) * vref *
0.0454; //46.45 / 1023 = 0.0454
 return volt;
}

float read_current(int
currentPin) {
 float vref = readVcc() / 1000;
```

```
 float aux = analogRead(currentPin) * vref / 1023;
 float azero = vref / 2;
 float curr = abs(aux - azero) / 0.066;
 return curr;
}

float real_power(int number_of_samples) {
 int n;
 float inst_power = 0;
 float sum_inst_power = 0;
 float inst_voltage;
 float inst_current;
 for(n = 0; n < number_of_samples; n++) {
  inst_voltage = read_voltage(voltagePin);
  inst_current = read_current(currentPin);
  inst_power = inst_voltage * inst_current;
  sum_inst_power += inst_power;
 }
 float real_power = sum_inst_power / number_of_samples;
 return real_power;
}

float rms_voltage(int number_os_samples) {
 int n;
 float squared_voltage = 0;
 float sum_squared_voltage = 0;
 float mean_square_voltage;
 float root_mean_square_voltage;
 float inst_voltage;
 for (n = 0; n < number_of_samples; n++) {
  inst_voltage = read_voltage(voltagePin);
  squared_voltage = inst_voltage * inst_voltage;
  sum_squared_voltage += squared_voltage;
 }
 mean_square_voltage = sum_squared_voltage / number_of_samples;
 root_mean_square_voltage = sqrt(mean_square_voltage);
 return root_mean_square_voltage;
}

float rms_current(int number_os_samples) {
 int n;
 float squared_current = 0;
 float sum_squared_current = 0;
 float mean_square_current;
 float root_mean_square_current;
 float inst_current;
 for (n = 0; n < number_of_samples; n++) {
  inst_current = read_current(currentPin);
  squared_current = inst_current * inst_current;
  sum_squared_current += squared_current;
 }
 mean_square_current = sum_squared_current / number_of_samples;
 root_mean_square_current = sqrt(mean_square_current);
 return root_mean_square_current;
}
```

Apêndice C – Código aplicação *Google Android*

index.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="css/topcoat-mobile-light.min.css" />
    <link rel="stylesheet" type="text/css" href="css/index.css" />
    <title>Wattímetro</title>
  </head>
  <body>
    <div class="topcoat-navigation-bar">
      <div class="topcoat-navigation-bar__item left quarter">
      <!--
        <a class="topcoat-icon-button--quiet">
          <span class="topcoat-icon topcoat-icon--menu-stack"></span>
        </a>
      -->
      </div>
      <div class="topcoat-navigation-bar__item center half">
        <h1 class="topcoat-navigation-bar__title">Wattímetro</h1>
      </div>
      <div class="topcoat-navigation-bar__item right quarter">
        <a class="topcoat-icon-button--quiet">
          <span class="topcoat-icon topcoat-icon--edit"></span>
        </a>
      </div>
    </div>

    <div id="content">

      <div id="testes" style="20px"></div>

      <div id="connection">
        <div style="margin: 20px">
          Dispositivo: <br/>
          <select id="deviceList">

<option>Buscando...</option>
          </select>
        </div>
        <p>
          <a class="topcoat-button--large is-disabled" id="connectButton">Conectar</a>
        </p>
        <p>
          <a class="topcoat-button--large" id="listButton">Listar Dispositivos</a>
        </p>
      </div>

      <div id="chat">
        <div id="messages" class="center" style="margin-top: 20px"></div>
        <div id="placeholder" style="width:270px;height:300px"></div>

        <div class="center" style="margin-top: 20px">
          <a class="topcoat-button" id="disconnectButton">Desconectar</a>
        </div>
      </div>
    <!--
      <div id="chat">
        <form id="chatform">
          <textarea id="messages" readonly></textarea>
          <input type="text" class="topcoat-text-input" id="message" placeholder="Message" />
          <a class="topcoat-button--cta" id="sendButton">Send</a>
        </form>
        <div class="center">
          <a class="topcoat-button" id="disconnectButton">Disconnect</a>
        </div>
      </div>
      -->
    </div>
    <footer>
      <div id="statusMessage"></div>
    </footer>
    <script type="text/javascript" src="cordova.js"></script>
    <script type="text/javascript" src="js/index.js"></script>
    <script language="javascript" type="text/javascript" src="flot/jquery.js"></script>
    <script language="javascript" type="text/javascript" src="flot/jquery.flot.js"></script>
    <script language="javascript" type="text/javascript" src="flot/jquery.flot.navigate.js"></script>
    <script type="text/javascript">
      app.initialize();
    </script>

    <script type="text/javascript">

      $(function() {

        // We use an inline data source in the example, usually data would
        // be fetched from a server

        var data = [],
          totalPoints = 250;
        function getData() {

          if (data.length > 0)
            data = data.slice(1);

          while (data.length < totalPoints) {

data.push(parseFloat(medicao));
          }

          // Zip the generated y values with the x values

          var res = [];
          for (var i = 0; i < data.length; ++i) {
```

```
                res.push([i,
data[i]])
            }

            return res;
        }

        // Set up the control
widget

        var updateInterval = 1;

        // setup plot
        var options = {
            series: { shadowSize:
0 }, // plota o gráfico sem
sombra para melhor
desempenho
            xaxis: { show: false },
            yaxis: { min: 0,
                panRange: [0,
1000]
            },
            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()]);

            // Since the axes don't
change, we don't need to call
plot.setupGrid()

            plot.draw();
            setTimeout(update,
updateInterval);
        }

        update();
    });

</script>

</body>
</html>
```

index.js

```
'use strict';

top.medicao = '0.00';
top.tensao = '0.00';
top.corrente = '0.00';
top.fp = '0.00';

var app = {
    initialize: function() {
        this.bind();
        listButton.style.display =
"none";
    },
    bind: function() {

document.addEventListener('de
viceready', this.deviceready,
false);
    },
    deviceready: function() {
        connectButton.ontouchstart
= app.connect;
        listButton.ontouchstart =
app.list;


disconnectButton.ontouchstart =
app.disconnect;


bluetoothSerial.subscribe("\n",
app.onmessage,
app.generateFailureFunction("S
ubscribe Failed"));

        setTimeout(app.list, 2000);
    },
    list: function(event) {

deviceList.firstChild.innerHTM
L = "Buscando...";

        app.setStatus("Procurando
por dispositivos Bluetooth...");

bluetoothSerial.list(app.ondevic
elist,
app.generateFailureFunction("Li
st Failed"));
    },
    connect: function() {
        var device =
deviceList[deviceList.selectedIn
dex].value;

app.disable(connectButton);

app.setStatus("Conectando...");
        console.log("Requisitando
conexão com " + device);

bluetoothSerial.connect(device,
app.onconnect,
app.ondisconnect);
    },
    disconnect: function(event) {
        if (event) {
            event.preventDefault();
        }


app.setStatus("Desconectando...
");

bluetoothSerial.disconnect(app.
ondisconnect);
    },
    ondevicelist:
function(devices) {
        var option;

        deviceList.innerHTML =
"";

        app.setStatus("");


devices.forEach(function(device
) {

            option =
document.createElement('option
');
            if
(device.hasOwnProperty("uuid"
)) {
                option.value =
device.uuid;
            } else if
(device.hasOwnProperty("addre
ss")) {
                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 =
"Nenhum dispositivo
Bluetooth";

deviceList.appendChild(option);
```

```
        if (cordova.platformId === "ios") { // BLE

app.setStatus("Nenhum periférico Bluetooth encontrado.");
        } else { // Android
            app.setStatus("Por favor pareie com um dispositivo Bluetooth.");
        }


app.disable(connectButton);
        listButton.style.display = "";
      } else {

app.enable(connectButton);
        listButton.style.display = "none";

app.setStatus("Encontrado" + (devices.length === 1 ? " " : "s") + devices.length + " dispositivo" + (devices.length === 1 ? "." : "s."));
      }

  },
  onconnect: function() {
      connection.style.display = "none";
      chat.style.display = "block";

app.setStatus("Conectado");
  },
  ondisconnect: function(reason) {
      var details = "";
      if (reason) {
          details += ": " + JSON.stringify(reason);
      }
      connection.style.display = "block";
      app.enable(connectButton);
      chat.style.display = "none";

app.setStatus("Desconectado");
  },
  onmessage: function(message) {
          var teste = message;
          var data = teste.split(',');
      tensao = data[0];
      corrente = data[1];
      fp = data[2];
      medicao = tensao * corrente;
      messages.innerHTML = medicao + " W";

      $.ajax({
        type: "POST",
        url: 'http://wattmeter-web.herokuapp.com/publish/',
        data: {
          'device': 'proto',
          'current': corrente,
          'voltage': tensao,
          'created': new Date($.now())
        },
        success: function (data) {
          if (data.success) {

$('#status').html('SUCCESS');
          } else {

$('#status').html('ERROR');
          }
        }
      });

  },
  setStatus: function(message) {
      console.log(message);


window.clearTimeout(app.statusTimeout);
      statusMessage.innerHTML = message;
      statusMessage.className = 'fadein';

      app.statusTimeout = setTimeout(function () {

statusMessage.className = 'fadeout';
      }, 5000);
  },
  enable: function(button) {
      button.className = button.className.replace(/\bis-disabled\b/g,'');
  },
  disable: function(button) {
      if (!button.className.match(/is-disabled/)) {
          button.className += " is-disabled";
      }
  },
  generateFailureFunction: function(message) {
      var func = function(reason) {
          var details = "";
          if (reason) {
              details += ": " + JSON.stringify(reason);
          }
          app.setStatus(message + details);
      };
      return func;
  }
};
```

Apêndice D – Código aplicação *web*

admin.py

```
from django.contrib import admin

from measures.models import Measure

class MeasureAdmin(admin.ModelAdmin):
    list_display = ('power', 'voltage', 'current', 'created')

    class Meta:
        model = Measure

admin.site.register(Measure, MeasureAdmin)
```

forms.py

```
from django import forms

from measures.models import Measure

class MeasureForm(forms.ModelForm):

    class Meta:
        model = Measure
```

models.py

```
from django.db import models

from accounts.models import Device

class Measure(models.Model):
    current = models.FloatField()
    voltage = models.FloatField()
    created = models.DateTimeField(auto_now=True, auto_now_add=True)
    device = models.ForeignKey(Device, blank=True, null=True, on_delete=models.SET_NULL)

    @property
    def power(self):
        return self.current * self.voltage

    class Meta:
        ordering = ['-created']
```

views.py

```
import csv
import json

from django.contrib.auth.decorators import login_required
from django.http import HttpResponse
from django.shortcuts import render
from django.views.decorators.csrf import csrf_exempt

from measures.forms import MeasureForm
from measures.models import Measure


@login_required
def index(request):
    measures = Measure.objects.filter(device__user=request.user).order_by('-created')[:250]
    measures = reversed(measures)
    return render(request, 'measures/index.html', {
        'measures': measures,
    })


@login_required
def export(request):
    response = HttpResponse(content_type='text/csv')
    response['Content-Disposition'] = 'attachment; filename="user_measures.csv"'

    writer = csv.writer(response)
    measures = Measure.objects.filter(device__user=request.user)

    for measure in measures:
        writer.writerow([measure.created, measure.current, measure.voltage, measure.power])

    return response


@csrf_exempt
def publish(request):
    result = {
        'success': False,
    }

    if request.method == 'POST':
        form = MeasureForm(request.POST)
        if form.is_valid():
            form.save()
            result['success'] = True
        print form.errors

    return HttpResponse(json.dumps(result), content_type='application/json')


def test(request):
    return render(request, 'measures/test.html')
```

urls.py

```
from django.conf.urls import patterns, include, url
from django.contrib import admin
from django.contrib.auth import views as auth_views

urlpatterns = patterns('',
    # Examples:
    url(r'^$', 'measures.views.index', name='measures_index'),
    url(r'^publish/$', 'measures.views.publish', name='measures_publish'),
    url(r'^measures/export/$', 'measures.views.export', name='measures_export'),
    url(r'^measures/test/$', 'measures.views.test', name='measures_test'),
    url(r'^accounts/devices/$', 'accounts.views.devices', name='accounts_devices'),

    url(r'^accounts/devices/turn_(?P<status>(on|off))/(?P<device_id>[\w]+)/$', 'accounts.views.turn_device', name='accounts_turn_device'),

    url(r'^accounts/devices/status/(?P<device_id>[\w]+)/$', 'accounts.views.device_status', name='accounts_device_status'),

    url(r'^accounts/get_device_id/$', 'accounts.views.get_device_id', name='accounts_get_device_id'),

    url(r'^accounts/get_device_username/$', 'accounts.views.get_device_username', name='accounts_get_device_username'),

    url(r'^accounts/login/$', auth_views.login, {'template_name': 'accounts/login.html'}, name='login'),
    url(r'^accounts/logout/$', auth_views.logout, {'next_page': '/'}, name='logout'),

    url(r'^admin/', include(admin.site.urls)),
)
```

base.html

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Wattímetro | {% block title %}Home{% endblock %}</title>
    <link href="{{ STATIC_URL }}css/normalize.css" rel="stylesheet" type="text/css">
    <link href="{{ STATIC_URL }}css/measures.css" rel="stylesheet" type="text/css">
    <script language="javascript" type="text/javascript" src="{{ STATIC_URL }}vendor/jquery.js"></script>
    {% block extra_head %}
    {% endblock %}
</head>
<body>
    <div class="container">
        <header class="clearfix">
            <h1>Wattímetro</h1>
            <nav>
                <ul>
                    {% if user.is_authenticated %}
                        <li><a href="{% url 'measures_index' %}">Gráfico</a></li>
                        <li><a href="{% url 'accounts_devices' %}">Leitores</a></li>
                        <li><a href="{% url 'measures_export' %}">Exportar</a></li>
                        <li><a href="{% url 'logout' %}">Sair</a></li>
                    {% endif %}
                </ul>
            </nav>
        </header>

        {% if messages %}
            <ul class="messages">
                {% for message in messages %}
                    <li{% if message.tags %} class="{{ message.tags }}"{% endif %}>{{ message }}</li>
                {% endfor %}
            </ul>
        {% endif %}

        <div class="main">
            {% block content %}{% endblock %}
        </div>
    </div>
</body>
</html>
```

login.html

```
{% extends 'base.html' %}

{% block title %}Home{% endblock %}

{% block content %}

    {% if form.errors %}
        <p>Nome de usuário ou senha errados.</p>
    {% endif %}

    <form method="post" action=".">
        {% csrf_token %}
        <table>
            <tr><td><label for="id_username">Nome de usuário:</label></td><td>{{ form.username }}</td></tr>
            <tr><td><label for="id_password">Senha:</label></td><td>{{ form.password }}</td></tr>
        </table>

        <input type="submit" value="login" />
        <input type="hidden" name="next" value="{{ next }}" />
    </form>

{% endblock %}
```

devices.html

```
{% extends 'base.html' %}

{% block title %}Leitores{% endblock %}

{% block content %}

   <table class="devices">
      <thead>
         <td>Leitor</td>
         <td>Ação</td>
      </thead>
      {% for device in devices %}
         <tr class="{% cycle 'odd' 'even' %}">
            <td>{{ device.id }}</td>
            <td>
               {% if device.turned_on %}
                  <a class="off" href="{% url 'accounts_turn_device' 'off' device.id %}">DESLIGAR</a>
               {% else %}
                  <a class="on" href="{% url 'accounts_turn_device' 'on' device.id %}">LIGAR</a>
               {% endif %}
            </td>
         </tr>
      {% endfor %}
   </table>

{% endblock %}
```

index.html

```
{% extends "base.html" %}

{% block title %}Home{% endblock %}

{% block extra_head %}
   <script language="javascript" type="text/javascript" src="{{ STATIC_URL }}vendor/jquery.flot.js"></script>
   <script type="text/javascript">

      $(function() {

         var powerMeasures = [];
         var voltageMeasures = [];
         var currentMeasures = [];
         {% for measure in measures %}

powerMeasures.push([{{ forloop.counter }}, {{ measure.power }}]);

voltageMeasures.push([{{ forloop.counter }}, {{ measure.voltage }}]);

currentMeasures.push([{{ forloop.counter }}, {{ measure.current }}]);
         {% endfor %}

         $.plot("#measures-graph", [ { data: powerMeasures, label: "Potência (W)" } ],{
            xaxes: [
               { position: 'bottom' }
            ],
            yaxes: [
               { position: 'left' }
            ],
               legend: { position: "se" }
         });
         $.plot("#volt-curr-graph", [ { data: voltageMeasures, label: "Tensão (V)" },
                                      { data: currentMeasures, label: "Corrente (A)", yaxis: 2 }
                                    ],{
            xaxes: [
               { position: 'bottom' }
            ],
            yaxes: [
               { position: 'left' },
               { position: 'right' }
            ],
               legend: { position: "se" }
         });
         //$.plot("#measures-graph", [ voltageMeasures ]);
         //$.plot("#measures-graph", [ currentMeasures ]);
      });

   </script>
{% endblock %}

{% block content %}

   <center><h1>Potência x Tempo</h1></center>
   <div id="measures-graph"></div>

<center><h1>Tensão/Corrente x Tempo</h1></center>
   <div id="volt-curr-graph"></div>
{% endblock %}
```

Apêndice E – Fluxograma completo do projeto