CENTRO UNIVERSITÁRIO DE BRASÍLIA - UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

MARCOS VINÍCIUS NERY

SOLUÇÃO PARA GESTÃO DO CONSUMO DE ENERGIA RESIDENCIAL

Orientador: Prof.ª. M.C. MARIA MARONY SOUSA FARIAS

Brasília

Novembro, 2013

MARCOS VINÍCIUS NERY

SOLUÇÃO PARA GESTÃO DO CONSUMO DE ENERGIA RESIDENCIAL

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientador: Prof.ª. M.C. Maria Marony Sousa Farias

Brasília

Novembro, 2013

MARCOS VINÍCIUS NERY

SOLUÇÃO PARA GESTÃO DO CONSUMO DE ENERGIA RESIDENCIAL

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientador: Prof.ª. M.C. Maria Marony Sousa Farias

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Prof. Abiezer Amarilia Fernandes
Coordenador do Curso

Banca Examinadora:

_____

Prof.ª. Maria Marony Sousa Farias
Mestre, Uniceub

_____

Prof.ª. Irene de Azevedo Lima Joffily
Mestre, Uniceub.

_____

Prof. Marco Antônio de Oliveira Araújo
Mestre, Uniceub

_____

Prof. Sidney Cerqueira Bispo dos Santos
Mestre, Uniceub

Agradeço em primeiro lugar a Deus por me direcionar e me abençoar. Aos meus pais pelo apoio e incentivo em toda a minha vida. À todos meus amigos pela motivação durante todos esses anos.

“O temor do Senhor ensina a sabedoria,
e a humildade antecede a honra.”
Provérbios 15:33.

Sumário

LISTA DE FIGURAS

LISTA DE EQUAÇÕES

RESUMO

Neste projeto é apresentada uma solução para gestão do consumo de energia em um ambiente residencial. O sistema proposto é composto de um micro controlador Arduino UNO, sensor de corrente, transformador de tensão, circuito integrador e interface Web. O protótipo é embarcado ao medidor de energia residencial, possibilitando ao usuário gerir de forma facilitada o consumo de energia. O projeto utiliza um sensor de corrente não invasivo, o que permite uma fácil montagem e instalação em qualquer circuito. Do sistema acoplado ao medidor é enviado via tecnologia *Bluetooth* todas as informações captadas a um servidor, onde estas informações são tratadas e armazenadas para serem disponibilizadas para os usuários via Web. O sistema é capaz de captar informações como: potência ativa, potência aparente, fator de potência, tensão, corrente e quilowatts-hora consumidos. Além destes dados é calculado, estimado e disponibilizado também com base nestas informações, o valor em reais que está sendo consumido periodicamente e uma previsão da conta de energia elétrica ou gastos realizados em determinado período.

Palavras-chave: Arduino, sensor de corrente, transformador de tensão, medidor de energia, energia elétrica, potência ativa, potência aparente, fator de potência, tensão, consumo, *Bluetooth*.

ABSTRACT

In this project is presented a solution of managing power consumption for residential environment. The proposed system consists of an Arduino UNO microcontroller, current sensor, voltage transformer, integrator circuit and Web interface. The prototype is embedded to residential power meter, enabling the user to easily manage energy consumption. The project uses a non-invasive sensor current, which allows easy assembly and installation in any circuit. To the coupled system, is sent by Bluetooth technology all captured information to a server, where the information is processed and stored to be available to users via Web. The system is able to capture information such as active power, apparent power, power factor, voltage, current and kilowatt - hours consumed. In addition to these data is calculated and also available based on this information an estimated real value being consumed periodically, and a prevision of the electricity bill or expenses incurred in a given period.

Key-Words: Arduino, current sensor, power adapter, energy meter, electric energy, real power, apparent power, power factor, voltage, consumption, Bluetooth.

CAPÍTULO 1 – INTRODUÇÃO

1.1 - Apresentação do Problema

Diante do crescimento contínuo do uso de energia elétrica principalmente em países em desenvolvimento como o Brasil, cada vez mais se faz necessário adotar medidas de controle, economia e monitoramento do uso de energia nos mais variados locais para que o desperdício de energia seja minimizado.

Segundo dados da EPE (Empresa de Pesquisa Energética), o consumo nacional de energia elétrica na rede somou 115.124 giga watts-hora (GWh) no terceiro trimestre de 2013, o que representou um crescimento de 3,9% sobre igual período de 2012. Os segmentos de consumo são divididos pela EPE em classes como Residencial, Comercial, Industrial e Outros conforme mostrado na figura 1.1.

BRASIL	2013		
	1º tri.	2º tri.	3º tri.
Total	**2,8%**	**3,0%**	**3,9%**
Residencial	6,5%	5,4%	6,9%
Comercial	6,3%	4,6%	5,3%
Industrial	-2,1%	1,1%	1,1%
Outros	5,2%	2,1%	5,1%

Figura 1.1 – Variação sobre igual período em 2012.
(Fonte: Empresa de Pesquisa Energética)

Analisando a figura 1.1, segundo a classificação em classes, o setor residencial foi o maior contribuinte para o aumento com acréscimo de 6,9% em relação a 2012. Ainda segundo a empresa, esse aumento neste setor se deve à expansão da posse e ao maior uso de eletrodomésticos nos domicílios.

Se comparado somente o 3º trimestre de 2013 em relação ao do ano anterior pode ser observado o aumento considerável em GWh conforme figura 1.2.

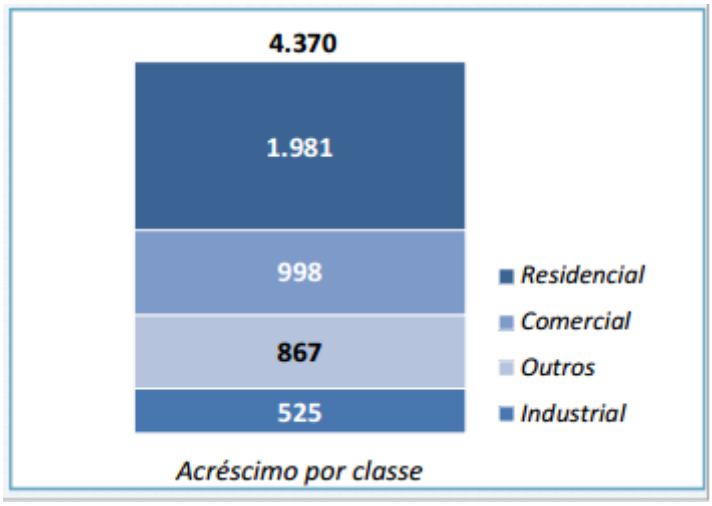

Figura 1.2 – Variação de GWh sobre igual período em 2012.
(Fonte: Empresa de Pesquisa Energética)

Considera-se o aumento então como um alerta contra o desperdício, a necessidade da economia e a melhor forma de utilização da energia elétrica.

No que diz respeito ao desperdício, um estudo realizado pela ABESCO (Associação Brasileira das Empresas de Serviços de Conservação de Energia), publicou que aproximadamente 10% do que o país consome atualmente é desperdiçado. Estes dados correspondem a mais do que o dobro de países como a Alemanha e representam para o país um desperdício de aproximadamente R$ 15 bilhões ao ano, conforme afirmou o presidente da ABESCO.

Uma das principais medidas governamentais adotadas para a economia é o horário de verão. Neste ano de 2013, segundo dados do ONS (Operador Nacional do Sistema Elétrico), a economia prevista é de R$400 milhões. E para isto serão 119 dias com horário adiantado.
Para economizar energia em ambientes residenciais, a ANEEL (Agência Nacional de Energia Elétrica) sugere diversas dicas de economia para que o consumidor faça melhor uso da energia, reduza o consumo e consequentemente diminua o valor na conta de luz. Dentre as dicas para o uso racional, encontra-se: Utilização natural da luz do dia, apagar as luzes ao deixar cômodos, utilizar eletrodomésticos que possuem o selo do PROCEL (Programa Nacional de Conservação de Energia Elétrica), calcular e conhecer consumo de seus aparelhos.

Apesar das sugestões apresentadas, há ainda a dificuldade de gerir o consumo diário em residências e o presente projeto surge como um adicional aos métodos utilizados e ao mesmo tempo uma alternativa ao acompanhamento do consumo de forma prática e clara ao consumidor.

Um ponto importante a se observar é que nem sempre o medidor de energia está em um local de fácil acesso ao consumidor, o que impede o acompanhamento do consumo. Além desta questão, as informações mostradas nos medidores convencionais não são tão claras para um usuário leigo e ainda que estejam em fácil acesso, exige que o usuário visualize, anote e calcule um valor estimado de consumo em relação à última leitura.

Desta forma, o sistema proposto permite um gerenciamento do consumo sem que seja necessário ir diretamente ao medidor de energia, bastando somente a interpretação dos dados disponibilizados via interface Web e sem necessidade de acompanhar seus gastos e a quantidade a ser paga num intervalo mensal somente após leitura e emissão da fatura para pagamento.

Além de não conseguir facilmente mensurar diariamente ou em intervalos customizados o que está sendo consumido, o cliente não obtém com facilidade uma estimativa parcial em reais que será paga por determinado período e nem consegue monitorar e identificar gastos indevidos causados por equipamentos defeituosos ou mal utilizados. Com base nesse contexto, fica a seguinte pergunta: Como gerir de forma mais prática e eficiente o consumo de energia e ter acesso a esta informação de forma alternativa à leitura usualmente feita no medidor de energia mensalmente?

1.2 - Objetivos do Trabalho

1.2.1 – Objetivo Geral:

O presente trabalho tem como objetivo principal projetar um sistema embarcado no medidor de energia residencial que não só identificará o consumo diário e cumulativo do consumo da casa como enviará estas informações a uma central onde essas possam ser tratadas, armazenadas e disponibilizadas para o usuário via Web. Na figura 1.3 é mostrada a topologia macro do projeto.

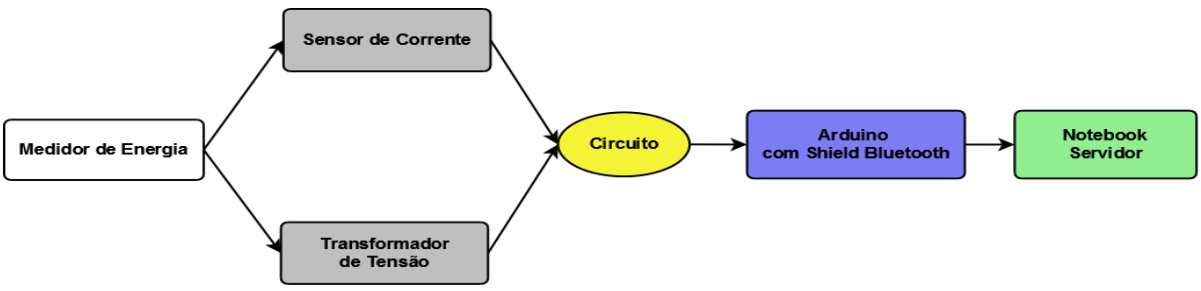

Figura 1.3 – Topologia macro do projeto. (Fonte: Autor)

1.2.2 – Objetivos Específicos:

Como objetivos específicos do projeto, temos:

- Monitoramento e armazenamento de informações sobre o consumo de energia elétrica como tensão, corrente, potência ativa e aparente, fator de potência e total de quilowatts-hora consumidos.
- Quantificação estimada em reais com base nas informações de consumo obtidos do sistema e com base na taxa de quilowatt-hora estabelecido pela companhia energética.
- Disponibilização de informações de consumo com base em datas definidas pelo usuário via WEB.

1.3 - Justificativa e Importância do Trabalho

Reconhecendo a dificuldade em gerenciar os gastos com energia em uma residência, empresas do ramo tecnológico estão investindo em tecnologias que auxiliem os consumidores a não só identificar a eficiência energética dos produtos como também gerir e quantificar monetariamente o consumo de seus equipamentos. Desta forma, o projeto de gestão do consumo de energia residencial visa auxiliar o consumidor na visualização do consumo diário e acumulado do gasto de energia de sua residência via WEB.

1.4 - Escopo do Trabalho

O projeto é realizado em um ambiente residencial onde se possa colher informações do medidor de energia, armazená-las e analisá-las via Web.

1.5 - Resultados Esperados

Espera-se que com a construção deste projeto seja possível a implementação de um protótipo medidor de energia com a utilização de sensor de corrente não invasivo e transformador de tensão. Espera-se ainda que ocorra sem falhas o estabelecimento de

comunicação *Bluetooth* entre protótipo medidor e o servidor para que ocorra a transmissão e gravação em banco de dados.

Por fim, espera-se otimizar o consumo de energia elétrica residencial, estimando o consumo de energia elétrica em Reais(R$), assim como gastos diários e acumulados estimados dentro de períodos customizados por meio de uma interface Web simplificada.

1.6 - Estrutura do Trabalho

O trabalho é apresentado da seguinte forma:

Capítulo 2: REFERENCIAL TEÓRICO

É apresentada a base teórica para entendimento do projeto e do modelo proposto assim como detalhamento e o que será utilizado no projeto.

Capítulo 3: SOLUÇÃO PARA GESTÃO DO CONSUMO DE ENERGIA RESIDENCIAL

É apresentado de fato o modelo proposto para a solução do problema apresentado no capítulo 1. Assim como apresentação do modelo será visto o detalhamento das etapas realizadas para obtenção da resolução do problema.

Capítulo 4: TESTES E RESULTADOS DA APLICAÇÃO DO MODELO

É apresentado de fato os testes implementados com o modelo proposto para a solução do problema apresentado no capítulo 1. Assim como os testes do modelo serão vistos os resultados obtidos.

Capítulo 5: APLICAÇÃO PRÁTICA DO MODELO PROPOSTO

Tem como objetivo apresentar uma aplicação prática do modelo, as dificuldades encontradas, a avaliação global e a análise do custo para implementação.

Capítulo 6: CONCLUSÃO

São apresentadas as conclusões obtidas com a realização deste projeto, assim como comentários sobre os resultados obtidos e características observadas. São apresentados também sugestões de continuação e melhorias a serem implementadas em novas versões que possam dar continuidade.

CAPÍTULO 2 – REFERENCIAL TEÓRICO

Para o desenvolvimento da solução do problema e entendimento do projeto em questão são necessários alguns conceitos. Este capítulo então descreve a teoria física necessária, os componentes utilizados para implementação e também as tecnologias e ferramentas utilizadas na concepção do projeto.

Dentre os conceitos teóricos envolvidos, os principais e alvos de medição do projeto são: energia, tensão, corrente elétrica e potência. Desta forma, estes são primeiramente definidos e relacionados com o projeto e posteriormente são descritos as tecnologias, os softwares e ferramentas utilizadas.

2.1– Tensão e Transformadores

Tensão é a quantidade de energia necessária para mover uma unidade de carga elétrica. Tensão (ou diferença de potencial) é a energia necessária para mover uma unidade de carga através de um elemento; é medida em volts (V). (ALEXANDER e SADIKU, 2003). Na prática é a quantidade de energia fornecida para movimentar uma carga elétrica em um condutor. É definida por:

$$V = \frac{W}{Q} \qquad \textbf{(3.1)}$$

Sendo que:

- V = Tensão em Volts (V);
- W = Quantidade de energia em Joules (J);
- Q = Carga elétrica em Coulombs (C).

Para obtenção da tensão elétrica que alimenta o local do experimento é utilizado um transformador que recebe em sua entrada a tensão da rede e fornece na saída uma tensão

contínua proporcional a entrada. Desta forma, obtém-se no protótipo, após redução da tensão no transformador e divisão de tensão no circuito a tensão que está sendo fornecida.

No geral, transformadores podem ser conceituados como equipamentos utilizados na transformação de tensão e corrente. Um transformador é um dispositivo usado para aumentar ou diminuir a tensão em um circuito sem uma perda de potência apreciável (TIPLER e MOSCA, 2006). Os transformadores possuem enrolamentos em torno de núcleos de materiais ferromagnéticos. Em um transformador simples, têm-se dois fios enrolados em um núcleo de ferro como ilustrado na figura 2.1.

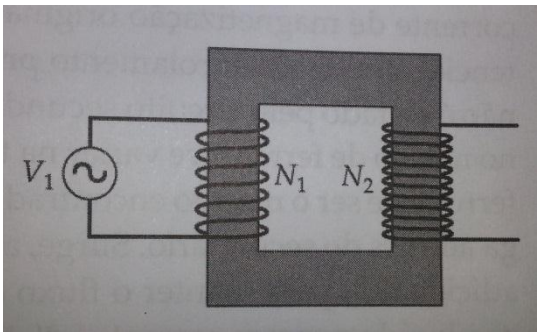

Figura 2.1 – Transformador com N_1 voltas no primário e N_2 no secundário. (Fonte: TIPLER, MOSCA, 2006)

O enrolamento transportando a potência de entrada é chamado de primário, e o outro enrolamento é chamado de secundário (TIPLER e MOSCA, 2006). O número de enrolamentos, ou seja, o número de espiras em cada enrolamento definirá o transformador como um transformador abaixador ou elevador de tensão. Didaticamente o número de voltas nos enrolamentos são denominados N_1 e N_2 se referindo aos enrolamentos do primário e do secundário respectivamente e V_1 e V_2 às diferenças de potenciais. A relação entre o número de espiras e as diferenças de potencial pode ser definida pela equação abaixo.

$$V_2 = \frac{N_2}{N_1} V_1 \tag{3.2}$$

Caso o número N_2 seja maior que N_1, ocorre que no enrolamento secundário terá uma diferença de potencial maior que no primário e se terá um transformador elevador. Quando ocorre o contrário, ou seja, N_1 é maior que N_2, ocorre que no enrolamento primário terá uma

diferença de potencial maior do que no secundário e se terá um transformador abaixador de tensão, que é o tipo de transformador utilizado para este projeto.

No protótipo foi utilizado um transformador capaz de reduzir a tensão de 220V para 9V + 9V com uma corrente de 1A. Segue imagem do equipamento utilizado na figura 3.2.

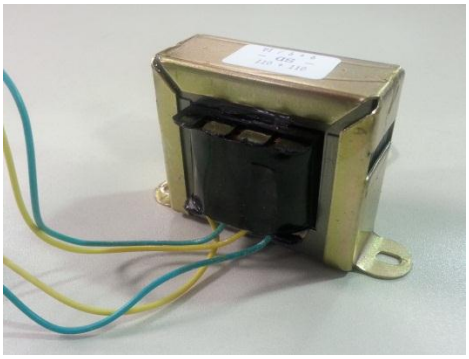

Figura 2.2 – Transformador Entrada 220V, com saída 9V + 9V.
(Fonte: Autor)

2.2 - Corrente elétrica e Sensor de corrente não invasivo

Corrente elétrica é definida como o fluxo de carga elétrica que passa por uma determinada área de seção transversal. Esse fluxo é provocado pela diferença de potencial (tensão). Quando uma chave é acionada, ligando um circuito, uma quantidade muito pequena de carga se acumula ao longo das superfícies dos fios e outros elementos condutores do circuito, e essas cargas produzem um campo elétrico que, no interior do material dos condutores, provoca o movimento de cargas moveis através dos materiais condutores presentes no circuito (TIPLER e MOSCA, 2006). A intensidade da corrente é definida como:

$$I = \frac{\Delta Q}{\Delta t} \tag{3.3}$$

Sendo que:

- I = Intensidade da corrente elétrica em Ampère (A);
- ΔQ = Carga que flui através de um condutor. Dada por Coulomb (C).
- Δt = Intervalo de tempo definido em segundos (s).

Pela fórmula de definição então se tem a divisão de Coulomb/segundos e por consequência a equivalência:

$$1A = 1C/s \tag{3.4}$$

Com base nos conceitos de tensão e corrente apresentados anteriormente, em resumo, temos que: A tensão aplicada é o mecanismo de partida; a corrente é uma reação à tensão aplicada (BOYLESTAD, 2012).

Para medição da corrente elétrica será utilizado um sensor de corrente não invasivo.

Os sensores de corrente não invasivos são sensores utilizados para medir corrente alternada. Para o projeto foi utilizado o sensor de modelo SCT013. Este sensor possui suporte a uma corrente de no máximo 100 Amperes e trabalha em temperaturas entre -25ºC e +70ºC. Pode ser clipado tanto na fase como no neutro do circuito que será possível captar a corrente passando pelo circuito em tempo real. Após captação, estes dados são enviados ao micro controlador Arduino UNO. Segue na figura 2.3 diagrama esquemático com especificações do sensor.

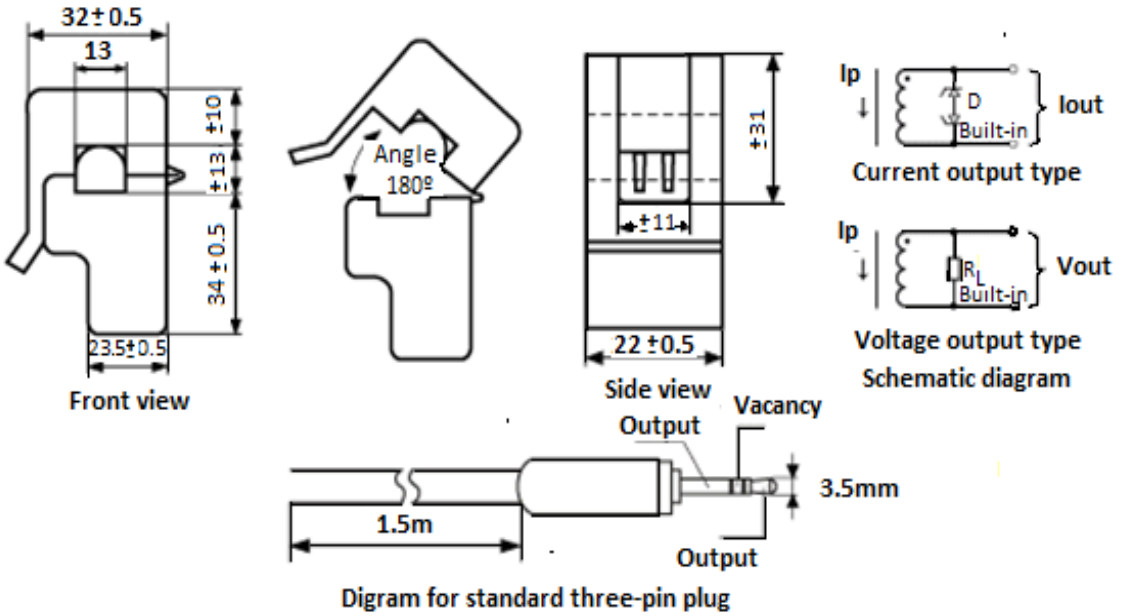

Figura 2.3 – Esquemático e dimensões do sensor de corrente não invasivo. (Fonte: http://www.seeedstudio.com/depot/images/product/2009511142810295.gif)

Para exemplificação, a figura 2.4 mostra a imagem do componente utilizado.

Figura 2.4 – Sensor de corrente não invasivo.
(Fonte: www.seeedstudio.com/depot/noninvasive-ac-current-sensor-100a-max-p-547.html)

2.3 - Energia elétrica e potência

Energia é a capacidade de realizar trabalho; é medida em joules (J) ou ainda em watt-segundo e Potência é a variação da energia (liberada ou absorvida) em função da variação do tempo; medida em watts (W) (ALEXANDER e SADIKU, 2003). Desta forma, Energia consumida ou fornecida pode ser definida como:

$$W = Pt \tag{3.5}$$

Sendo que:

- W = Energia (W);
- P = Potência (watt-segundo ou j/s);
- t = Segundos (s);

A potência pode ser definida por:

$$P = VI \tag{3.6}$$

Sendo que:

- P = Potência instantânea (watts);
- V = Tensão (Volts);
- I = Corrente (ampere);

Considerando a Lei de Ohm, mostrada na equação 7:

$$V = IR \tag{3.7}$$

Sendo que:

- V = Tensão em Volts (V);
- I = corrente em Amperes (A);
- R = Resistência (ohms);

E associando a definição de potencia demonstrada na equação (6), tem-se por substituição das variáveis que:

$$P = \frac{V^2}{R} \qquad \text{ou} \qquad P = I^2R \tag{3.8}$$

A potência pode ser classificada ainda em potência ativa, potência reativa e aparente.

A potência ativa, ou potência média segundo é o valor médio da potência instantânea durante um período. É a potência convertida da forma elétrica para formas não elétricas e vice-versa(NIELSON e RIEDEL, 2012). A potência média é expressa como:

$$P = UI.\cos(\theta_v - \theta_i) \tag{3.9}$$

Sendo que:

- P = Potência ativa em Watts (W);
- U = Tensão em Volts (V);
- I = Corrente elétrica em Ampères (A);
- θ_v= ângulo de fase da tensão;
- θ_i = ângulo de fase da corrente elétrica;

A potência reativa é a potência elétrica trocada entre o campo magnético de um indutor e a fonte que o alimenta ou entre o campo elétrico de um capacitor e a fonte que o

alimenta. A potência reativa nunca é convertida em potência não-elétrica (NIELSON e RIEDEL, 2012). A potência reativa é expressa como:

$$Q = UI.\text{sen}(\theta_v - \theta_i) \tag{3.10}$$

Sendo que:

- Q = Potência reativa (Var);
- U= Tensão em Volts (V);
- I = Corrente elétrica em Ampéres (A);
- θ_v= ângulo de fase da tensão;
- θ_i = ângulo de fase da corrente elétrica;

A potência aparente é o módulo da soma quadrática de potência ativa e reativa, dada em Volt-Ampere (VA) (NIELSON e RIEDEL, 2012). A potência aparente é expressa como:

$$|S| = \sqrt{P^2 + Q^2} \tag{3.11}$$

Sendo que:

- S = Potência aparente (VA);
- P = Potência ativa em Watts (W);
- Q = Potência reativa (Var);

Inicialmente, na equação 5, a potência está dada em watt-segundo. Para fins práticos, e por watt-segundo ser uma unidade pequena, são utilizados com mais frequência o Wh (watt-hora) e o (kWh) kilowatt-hora. Tem-se, portanto que:

$$Energia\ (Wh) = potencia(W)\ x\ tempo(h) \tag{3.12}$$

E que:

$$Energia\ (kWh) = \frac{potencia(W)\ x\ tempo}{1.000}\ (h) \tag{3.13}$$

A quantidade de Energia em kWh é o principal fator no cálculo do consumo de energia. Com base nesse dado, é possível verificar o consumo, por exemplo, em uma residência em determinado período. Na figura (2.5) é possível verificar um modelo de conta de energia e no campo delimitado pelo quadro preto, a quantidade de kWh consumidos.

Figura 2.5 – Modelo de conta de energia.
(Fonte: http://www.ceb.com.br/index.php/agencia-online, adaptada)

Para verificação final em Reais (R$) da conta, ainda é necessário multiplicar esta quantidade de kWh pelo valor da tarifa vigente da concessionária da região e adicionar a taxa de contribuição de iluminação pública. Segundo dados da ANEEL, o valor estipulado do kWh vigente de 26/08/2013 a 25/08/2014 para a CEB (Companhia Energética de Brasília), local da implementação do protótipo, é de R$ 0,25647 para a classe de consumo Residencial conforme figura 2.6.

Descrição	R$/kWh*
B1 - Residencial	0,25647
B1 - Residencial Baixa Renda	
Consumo mensal inferior ou igual a 30 kWh	0,08746
Consumo mensal superior a 30 kWh e inferior ou igual a 100 kWh	0,14993
Consumo mensal superior a 100 kWh e inferior ou igual a 220 kWh	0,22490
Consumo mensal superior a 220 kWh	0,24989

Figura 2.6 – Tarifa vigente para classe de consumo Residencial.
(Fonte: ANEEL)

É importante ressaltar que os valores da figura 3.6 se referem às tarifas homologadas pela ANEEL, expressas na unidade R$/kWh (reais por quilowatt-hora) e não contemplam

tributos e outros elementos que fazem parte da conta de luz, tais como: PIS, COFINS, ICMS e Taxa de Iluminação Pública. Estas definições são importantes para comprovação dos resultados obtidos e para cálculos de verificação realizados durante a implementação. No protótipo então, é utilizado como base o valor de tarifa homologado pela ANEEL, R$ 0,25647.

2.4 - Tecnologia Bluetooth e *Shield* Bluetooth

Bluetooth é uma forma de comunicação sem fio que permite a transmissão e comunicação de dados entre dispositivos que estejam próximos de uma forma rápida e com baixo consumo de energia. A transferência de dados é feita por radiofrequência e os dispositivos vão se detectando desde que estejam dentro do limite de distância suportado pela tecnologia. Segundo o site www.bluetooth.com, a tecnologia possui 3 diferentes classes. Sendo elas:

- Classe 3: Possui alcance de até 1 metro.
- Classe 2: Mais comum em dispositivos móveis. Possui alcance de até 10 metros.
- Classe 1: usado principalmente na indústria. Possui alcance de até 100 metros.

No presente projeto a comunicação *Bluetooth* utilizada pertence à classe 2.

Bluetooth é o nome da tecnologia WLAN desenvolvida para conectar dispositivos de diferentes funcionalidades como telefones, notebooks, computadores, câmeras, impressoras, cafeteiras e assim por diante. Uma LAN *Bluetooth* é uma rede ad hoc formada espontaneamente, isto é, os dispositivos, às vezes denominados *gadgets* (equipamentos eletrônicos em geral, pequenos e modernos), localizam uns aos outros e estabelecem uma rede denominada *piconet*. Uma LAN *Bluetooth* pode até mesmo se conectar à Internet, se um dos *gadgets* tiver essa capacidade. Por natureza, uma LAN *Bluetooth* não pode ser grande (FOROUZAN, 2006).

O *Bluetooth* pode ser utilizado para as mais diversas aplicações e se encontra nos mais variados dispositivos e periféricos de computador como mouses e teclados, sensores de segurança residencial e outros. A arquitetura *Bluetooth* define dois tipos de redes: *piconets* e *scatternet*.

Na *piconet* podem ter até 8 dispositivos sendo que um deles é a estação mestre e as demais são chamadas de estações escravas e a comunicação entre os dispositivos pode ser ponto a ponto ou multiponto. A figura 2.7 mostra a estrutura de uma rede *piconet*.

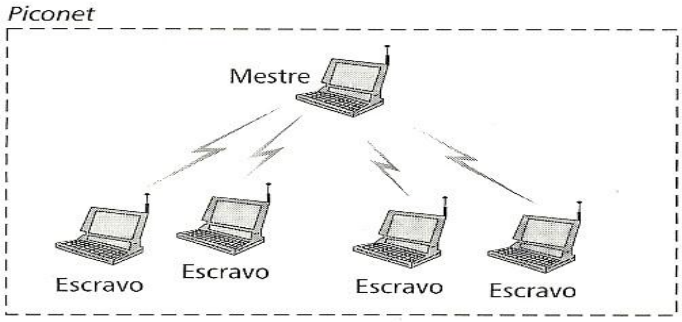

Figura 2.7 – Rede *Piconet*.
(Fonte: FOROUZAN, 2006)

Uma *scatternet* é a combinação de *piconets*, onde um dispositivo considerado escravo em uma rede *Bluetooth* se torna o dispositivo *master* numa segunda rede.

As piconets podem ser combinadas de modo a formar uma scatternet. Uma estação escrava numa piconet pode tornar-se a mestre noutra piconet. Esta estação pode receber mensagens do mestre na primeira *piconet* (como escrava) e agir como mestre repassando-as às escravas na segunda piconet. Uma estação pode pertencer simultaneamente a duas piconets (FOROUZAN, 2006). A figura 2.8 mostra uma rede *scatternet*.

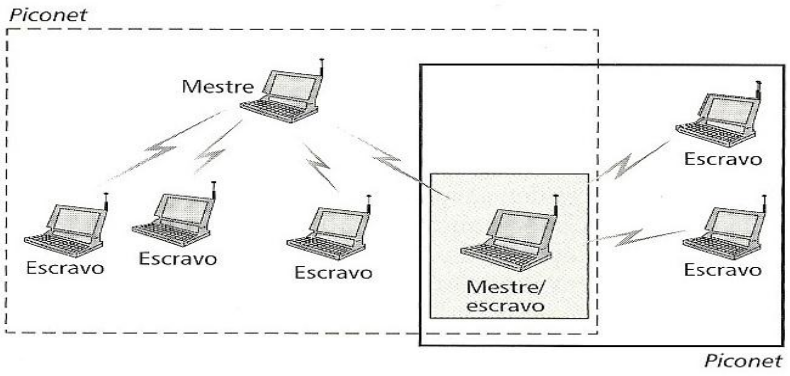

Figura 2.8 – Rede *Scatternet*.
(Fonte: FOROUZAN, 2006)

No projeto é utilizada uma rede do tipo *Piconet* entre o notebook servidor e o *shield Bluetooth* integrado ao micro controlador Arduino Uno. Ou seja, o notebook servidor é utilizado como estação master e o *Shield Bluetooth* é definido como estação escrava da rede.

O *shield Bluetooth*, como o nome sugere, possui um módulo de *Bluetooth* integrado e pode ser utilizado facilmente com um micro controlador como o Arduino utilizando a biblioteca *SoftwareSerial.h*, permitindo uma comunicação serial sem fio. Além da compatibilidade com o Arduino, o *Shield Bluetooth* possui alcance de 10m em área aberta e antena integrada ao *Shield*. Na figura 3.9 é mostrada a imagem do modelo utilizado.

Figura 2.9 – *Shield Bluetooth.*
(Fonte: http://www.seeedstudio.com/depot/bluetooth-shield-p-866.html?cPath=19_21)

Para melhor detalhamento, na figura 3.10, são mostradas as especificações do *Shield Bluetooth*:

Item	Min	Typical	Max	Unit
Voltage	2.8	3.3	3.5	VDC
Current	3	/	100	mA
Communication Distance(in house)	/	/	10	m
Protocol	Bluetooth V2.0 with SPP firmware			/
Interface	Uart Serial Port(TTL)			/
Supported Baudrate	9600, 19200, 38400, 57600, 115200, 230400, 460800			bps
ESD contact discharge	±4			KV
ESD air discharge	±8			/
Dimension	57.4x45.3x19.4			mm
Net Weight	10±2			g

Figura 2.10 – Especificações do *Shield Bluetooth*
(Fonte: http://www.seeedstudio.com/wiki/Bluetooth_Shield)

2.5 – Micro controlador Arduino UNO

Para a realização do projeto foi escolhido o Arduino UNO. Além de ser facilmente programado e possuir uma IDE(Integrated Development Enviroment) simples apresenta um custo benefício bom para este tipo de projeto. Arduino é definido como um pequeno computador que você pode programar para processar entradas e saídas entre o dispositivo e os componentes externos conectados a ele. O Arduino é o que chamamos de plataforma de computação física ou embarcada, ou seja, um sistema que pode interagir com seu ambiente por meio de hardware e software (MCROBERTS, 2011).

A placa de Arduino possui um micro controlador ATmega328, que possui 32 KB de memória, sendo 0.5 KB utilizado para *bootloader*. Tem ainda 2 KB de SRAM e 1 KB de EEPROM que pode ser utilizada para leitura e escrita. A placa trabalha com uma tensão de 5 volts, possui 14 entradas digitais, 6 analógicas e um botão reset para reiniciar o micro controlador. Pode ser alimentado via USB ou uma por fonte externa utilizando o conector jack.

Na figura 2.11 é ilustrado o Arduino UNO R3.

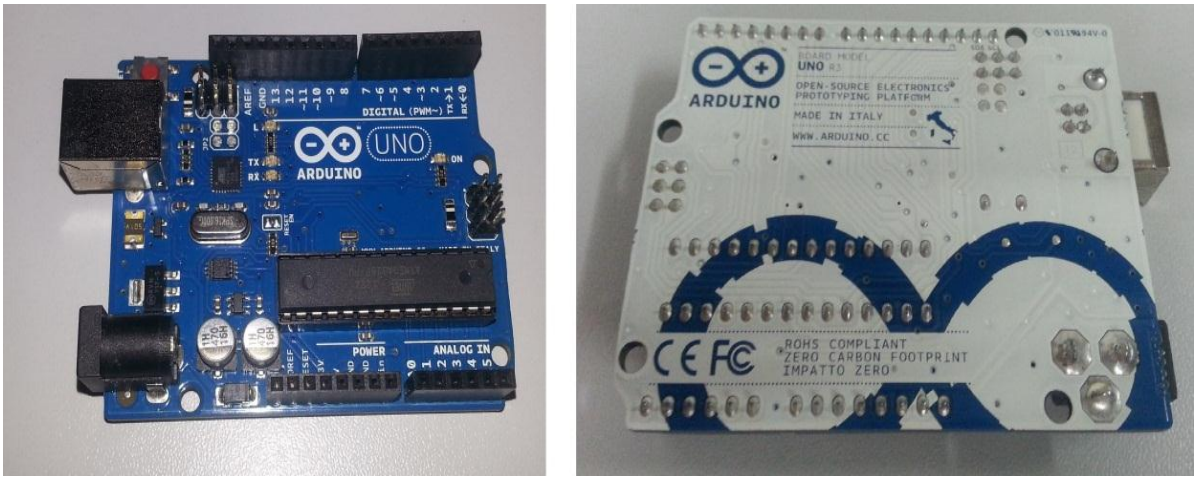

Figura 2.11 – Micro controlador Arduino UNO.
(Fonte: Autor)

Para melhor descrição e detalhamento, na figura 2.12 são mostradas as especificações da placa.

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage	7-12V

(recommended)	
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Figura 2.12 – Especificações do micro controlador Arduino UNO.
(Fonte: http://arduino.cc/en/Main/ArduinoBoardUno)

O Arduino, acoplado ao *Shield Bluetooth*, ao circuito utilizado para integrar o sensor de corrente e o transformador fornecedor da tensão, que serão apresentados posteriormente, captará informações como corrente e tensão que serão trabalhadas da maneira adequada no protótipo.

Para a interação entre o micro controlador Arduino Uno e um desktop, notebook ou outro dispositivo similar, é utilizado o Arduino Software, programa que será abordado no item 2.7 – Software.

2.6 - Protoboard MP 830

Para montagem dos circuitos foi utilizado neste projeto a proboard MP 830. A placa possui 830 furos, com dimensões 165(A) x 54(L) x 10(P)mm e massa de 103g. Possui suporte a tensão máxima de 300V RMS e corrente máxima de 3A RMS. A figura 2.13 ilustra imagem da *protoboard* utilizada.

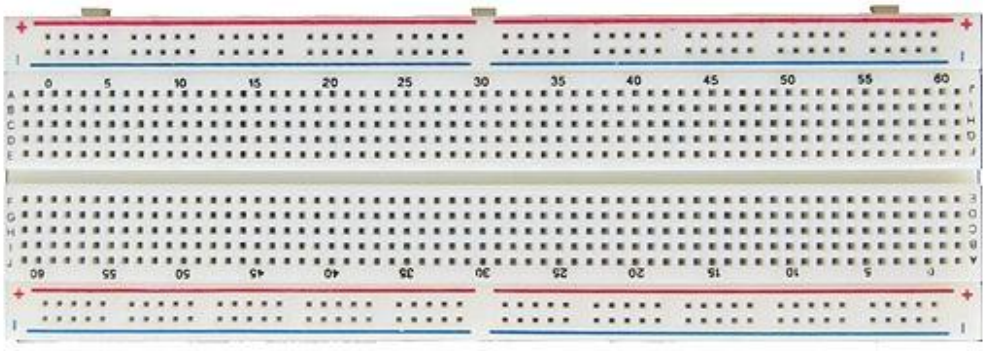

Figura 2.13 – Protoboard MP 830
(Fonte: http://www.minipa.com.br)

2.7 - Software

2.7.1 – Arduino Software

Arduino software é um programa *open-source* utilizado para codificar e carregar arquivos fontes para o micro controlador Arduino. Sendo que a linguagem do Arduino é baseada em C/C++. A linguagem de programação utilizada para criar código do Arduino é um dialeto da Linguagem C. (MCROBERTS, 2011). Na Figura 2.14 é mostrada a imagem do software.

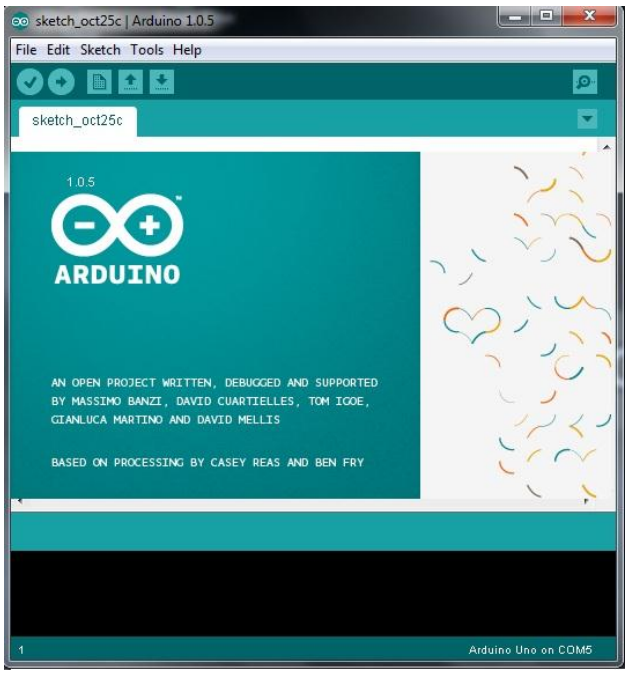

Figura 2.14 – IDE do Arduino e janela de informações e versão do software.
(Fonte: Autor)

Esta IDE possui um editor de texto para a realização da codificação, uma área de console, uma de demonstração de mensagens e barra de ferramentas com diversas funções. A aplicação pode ser executada em Sistemas Operacionais Windows, Mac OS X e Linux e possui interface simples e intuitiva. Para o projeto foi utilizada a versão 1.05 do software como mostrada na figura e se encontra disponível para download em http://arduino.cc/en/Main/Software. Utilizando o Arduino Software e um cabo USB é possível programar e enviar as informações via porta COM, para o micro controlador.

CAPÍTULO 3 – SOLUÇÃO PARA GESTÃO DO CONSUMO DE ENERGIA RESIDENCIAL

3.1 - Apresentação Geral do Modelo Proposto

O modelo geral do projeto engloba os principais itens do trabalho. Portanto é mostrada na Figura 3.1 imagem descritiva para entendimento da estrutura do projeto. O Sensor de corrente não invasivo mais o transformador abaixador de tensão colhem as informações de tensão, corrente e informações de potências que passam por um circuito elétrico e chegam ao micro controlador Arduino. Do Arduino são enviados os dados para o notebook servidor via conexão *Bluetooth* para que os dados sejam tratados, armazenados e disponibilizados para o usuário.

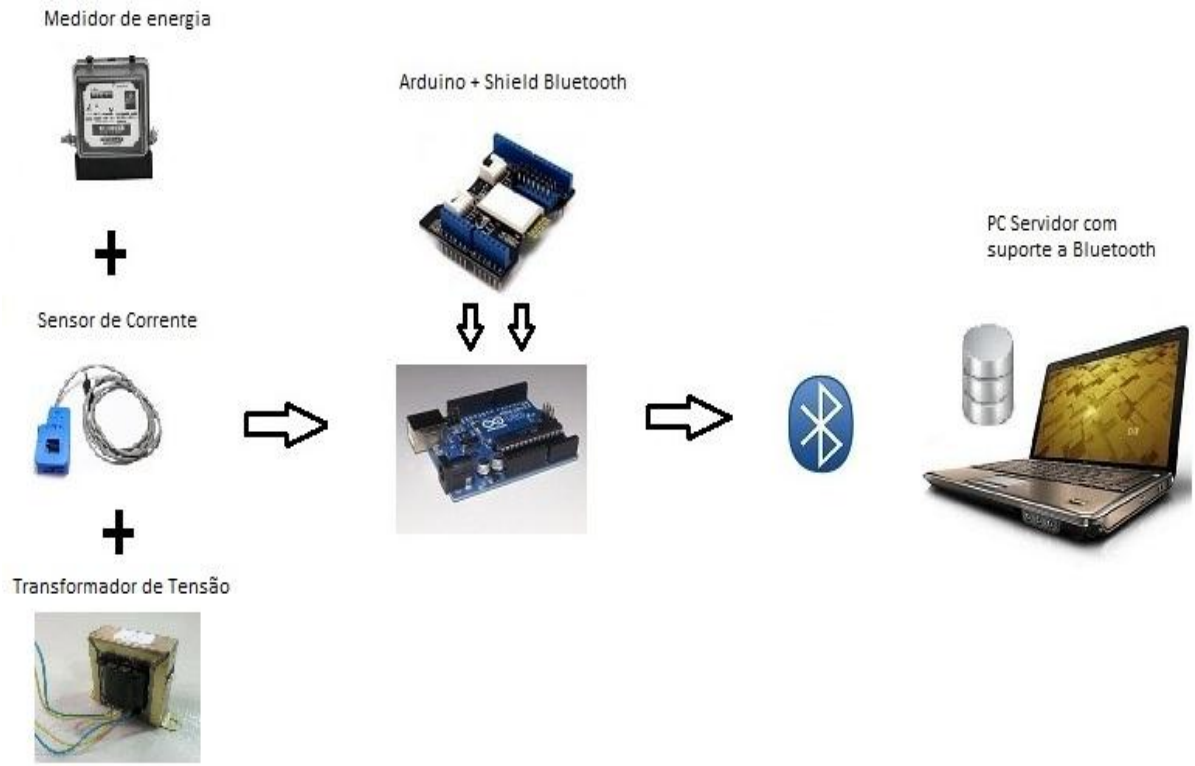

Figura 3.1 – Visão Geral do Modelo Proposto.
(Fonte: Autor)

3.2 - Descrição das Etapas do Modelo

Neste tópico são mostradas as etapas de construção do protótipo do início ao fim do processo com a finalidade de detalhar e explicar os procedimentos realizados e a função de cada fase. O funcionamento adequado do protótipo dependia de dois fatores básicos: A comunicação do Arduino integrado ao *Shield Bluetooth* com o Notebook Servidor e da leitura correta da tensão, corrente e informações de potência. Estas duas etapas iniciais foram então separadas em: etapa de montagem do *Shield Bluetooth*, integração com Arduino e testes de comunicação e etapa de construção do circuito e validação das medições. A junção das duas etapas então consistiu na leitura dos dados coletados e a transmissão destes por meio do *Bluetooth*. Posteriormente foi realizada a etapa de gravação em banco de dados e consulta do que foi registrado.

3.2.1 - Montagem do *Shield Bluetooth*, integração com Arduino e testes de comunicação

A primeira fase do processo de montagem de hardware constituiu da montagem do S*hield Bluetooth*, visto que seus barramentos e pinos vêm de fábrica desconectados e não soldados. Após soldar os barramentos adequadamente conforme documentação do fabricante foi possível empilhar o S*hield* na placa do Arduino conforme Figura 3.2.

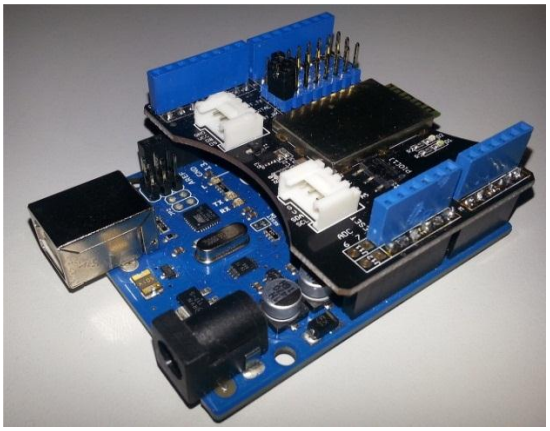

Figura 3.2 – Arduino UNO + *Shield Bluetooth*.
(Fonte: Autor)

Após integração física Arduino + *Shield Bluetooth,* o primeiro passo foi testar a comunicação *Bluetooth* com outros dispositivos com suporte a tecnologia para verificação da comunicação. Para o funcionamento do *Shield,* foi necessário a inclusão de duas bibliotecas, *BluetoothShieldDemoCode* e *NewSoftSerial*, que foram encontradas para download nas instruções de uso contidas na documentação do fabricante. Os códigos de exemplo do fabricante foram então adaptados e comentados conforme necessidade do projeto e por fim importados na IDE do Arduino para o micro controlador. O código para teste de comunicação, Código 1 – Código de comunicação *Bluetooth*, pode ser consultado no Apêndice.

Para a realização do teste de pareamento e comunicação foram utilizados um notebook com suporte a *Bluetooth* e posteriormente um celular com sistema operacional *Android* versão 4.1.2, também com suporte a *Bluetooth*. No notebook, para a realização dos testes foi utilizado o software SSCOM 3.2.

Na comunicação entre o Smartphone e o Arduino com *Shield Bluetooth* foram enviadas mensagens de ambas as partes e o destinatário recebeu as informações com sucesso. Para enviar e receber mensagens, o Arduino foi ligado à porta USB do notebook de modo a utilizar a porta COM 5 e obter também a alimentação necessária. Foi aberta então a conexão na porta 5 pelo SSCOM e aguardou-se a conexão via Smartphone conforme Figura 3.3.

Figura 3.3 – *Shield Bluetooth* aguardando comunicação com Smartphone (Fonte: Autor)

No Smartphone foi utilizado o aplicativo *Bluetooth* SPP, versão 1.991, que havia sido previamente instalado. O *Shield*, nomeado como "Shield_Nery", no código de comunicação, foi localizado pelo aparelho celular e a comunicação pode ser realizada conforme mostrado na figura 3.4.

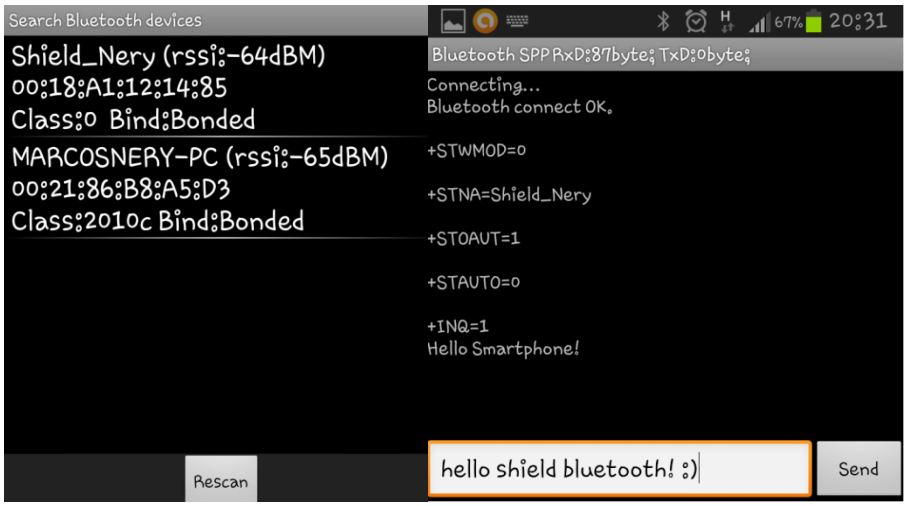

Figura 3.4 – *Smarphone* Comunicando com *Shield Bluetooth* e recebendo mensagem. (Fonte: Autor)

Após sincronização, o *Shield* também recebeu as mensagens do aparelho celular conforme figura 3.5.

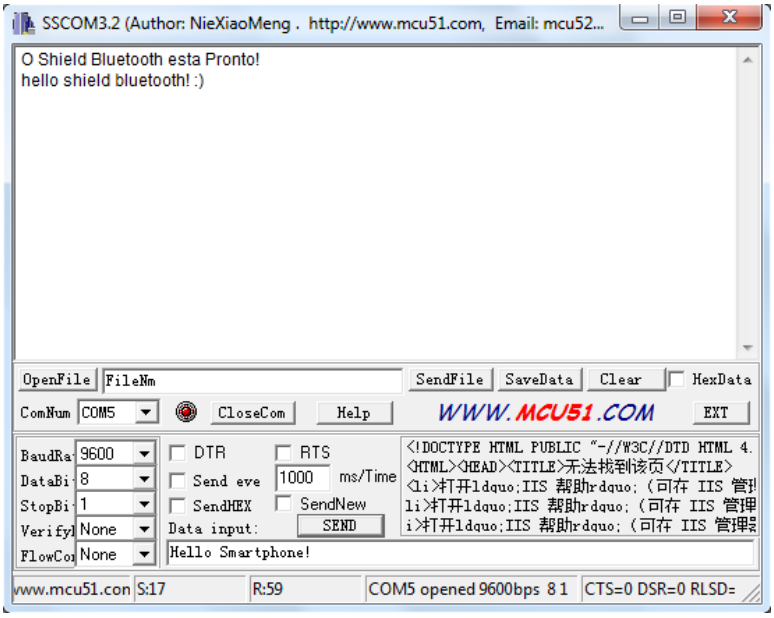

Figura 3.5 – *Shield Bluetooth* recebendo mensagem do Smartphone. (Fonte: Autor)

Na sincronização com o notebook, o processo foi semelhante ao procedimento realizado no pareamento com smartphone. Nesta etapa foi aberta uma instância do SSCOM para servir de interface para o notebook e outra como interface do *Shield Bluetooth* nas portas COM 8 e COM 5 respectivamente. A figura 3.6 mostra a comunicação descrita.

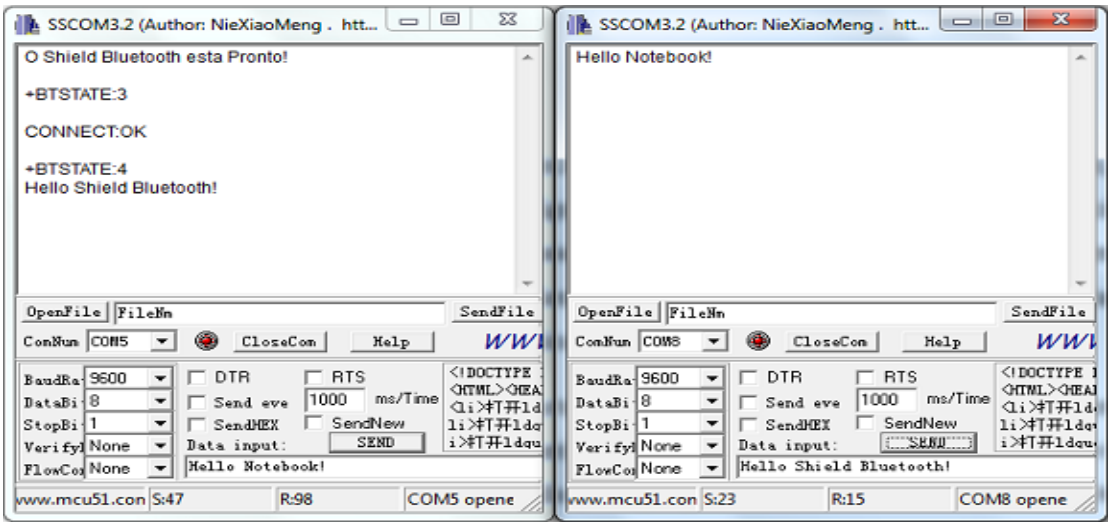

Figura 3.6 – *Shield Bluetooth* comunicando com Notebook.
(Fonte: Autor)

3.2.2 - Construção do circuito e validação das medições

Com a fase de comunicação concluída com sucesso, foi possível iniciar a segunda fase. Foi construído então o circuito para integração do sensor de corrente, transformador de tensão e o Arduino UNO. Na Figura 3.7 é ilustrado o diagrama elétrico do circuito montado para que os periféricos captassem e repassassem as informações ao micro controlador.

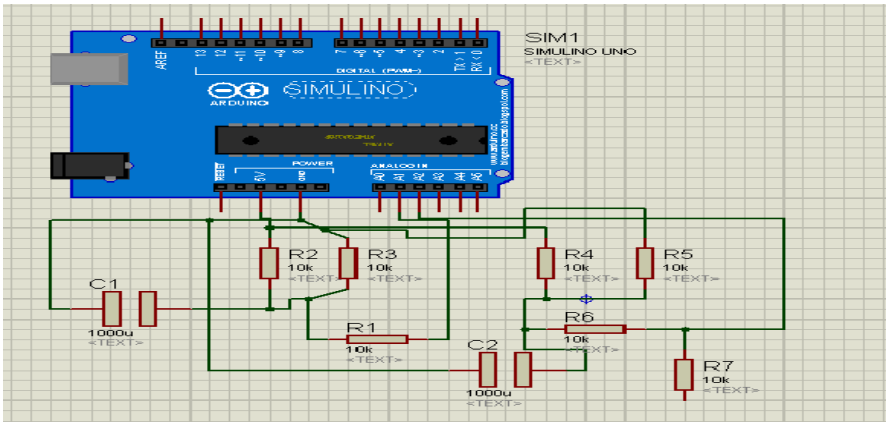

Figura 3.7 – Diagrama elétrico do Circuito construído no Proteus.
(Fonte: Autor)

Após confecção do projeto elétrico, montou-se o protótipo completo conforme figura 3.8.

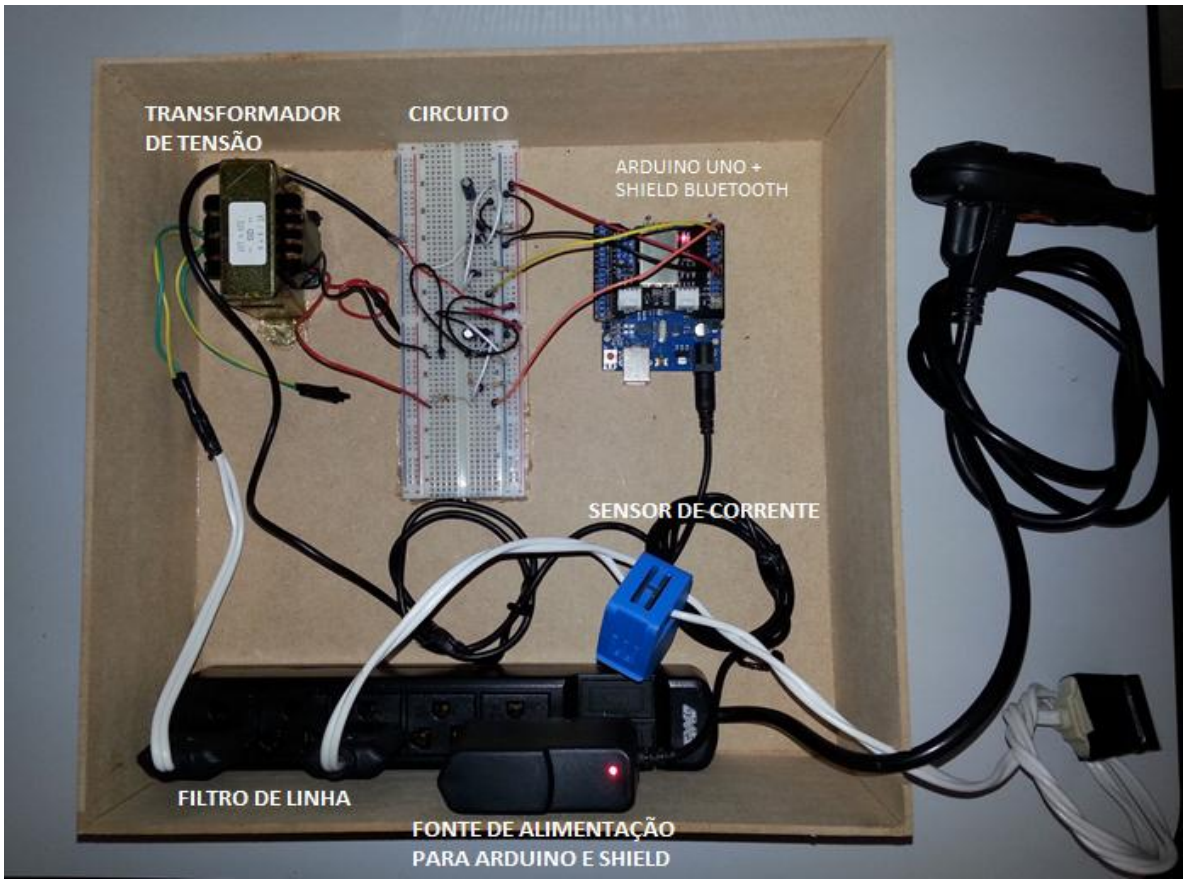

Figura 3.8 – Circuito Completo com Arduino, Transformador e sensor de Corrente. (Fonte: Autor)

Na figura, o sensor de corrente envolve o fio da fase da alimentação do circuito, ao passo que o transformador de energia fornece tensão em um nível aceitável para a entrada analógica do Arduino conforme explicado no capítulo 2.

Com a montagem do protótipo completa, foi implementada a parte de software para testes. Para o tratamento das informações obtidas no Arduino foi utilizada a biblioteca *Emonlib.h* e o código utilizado para obtenção dos dados se encontra no Apêndice, Código 2, sendo que no teste de tensão havia somente o cálculo de tensão e no código 2 citado encontra-se cálculo de tensão, corrente e potências.

Após Upload do código 2 foram realizados testes de leitura com uma lâmpada incandescente de 100W alternando seu status de ligado para desligado e observando os valores fornecidos. Para validação dos resultados de corrente, tensão e potência foram

calculados com base na tensão média encontrada pelo multímetro e as equações de potência definidas na seção 2.3, os valores teóricos esperados para o padrão da lâmpada utilizada nos testes.

Comparando os resultados fornecidos pelo programa e a tensão obtida pelo multímetro, foi feita a calibragem do software do protótipo. Neste processo, foram ajustados os parâmetros das funções "*voltage*" e "*current*" da instância "*emon*" conforme figura 3.9 até chegar aos valores de leitura esperados.

```
1  void setup()
2  {
3    Serial.begin(9600);
4
5    //definicao de entrada e saida bluetooth.
6    pinMode(RxD, INPUT);
7    pinMode(TxD, OUTPUT);
8
9    //passagem de parametros de configuracao para a instancia emon.
10   emon1.voltage(3, 255.26, 1.7);  // Voltage: input pin, calibration, phase_shift
11   emon1.current(2, 60.6);       // Current: input pin, calibration.
12
13   //chama funcao de configuração do shield bluetooth.
14   setupBlueToothConnection();
15 }
16
```

Figura 3.9 – Calibração das funções de cálculo de tensão e corrente.
(Fonte: Autor)

Realizada a calibração, à medida que foram efetuadas as leituras de tensão pelo multímetro, o valor foi comparado aos valores captados pelo protótipo montado conforme ilustrado na figura 3.10. A precisão da medição de tensão foi bem apurada visto que a diferença dada entre os valores visualizados no multímetro e os valores fornecidos pelo protótipo, se diferem apenas em casas decimais.

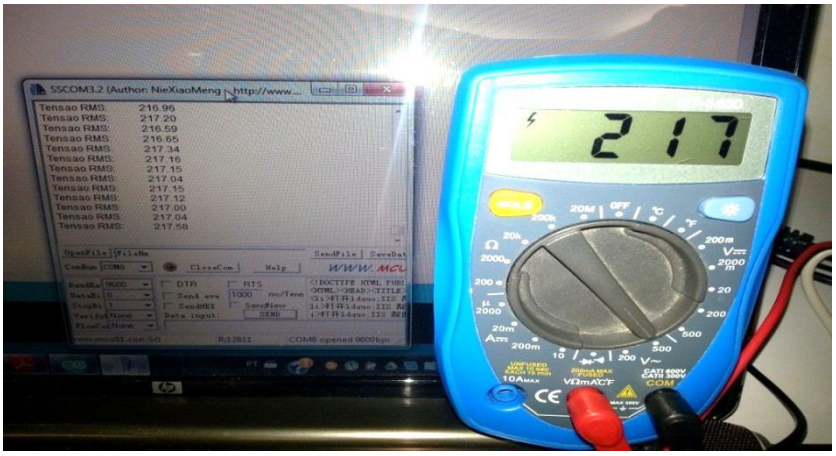

Figura 3.10 – Comparativo na leitura de tensão em tempo real.
(Fonte: Autor)

Analisando a figura 3.10 é possível observar que a tensão medida pelo protótipo foi fiel à medição realizada pelo multímetro dado a calibração realizada.

Verificada a medição da tensão foi verificada a leitura dos demais dados conforme mostra a figura 3.11.

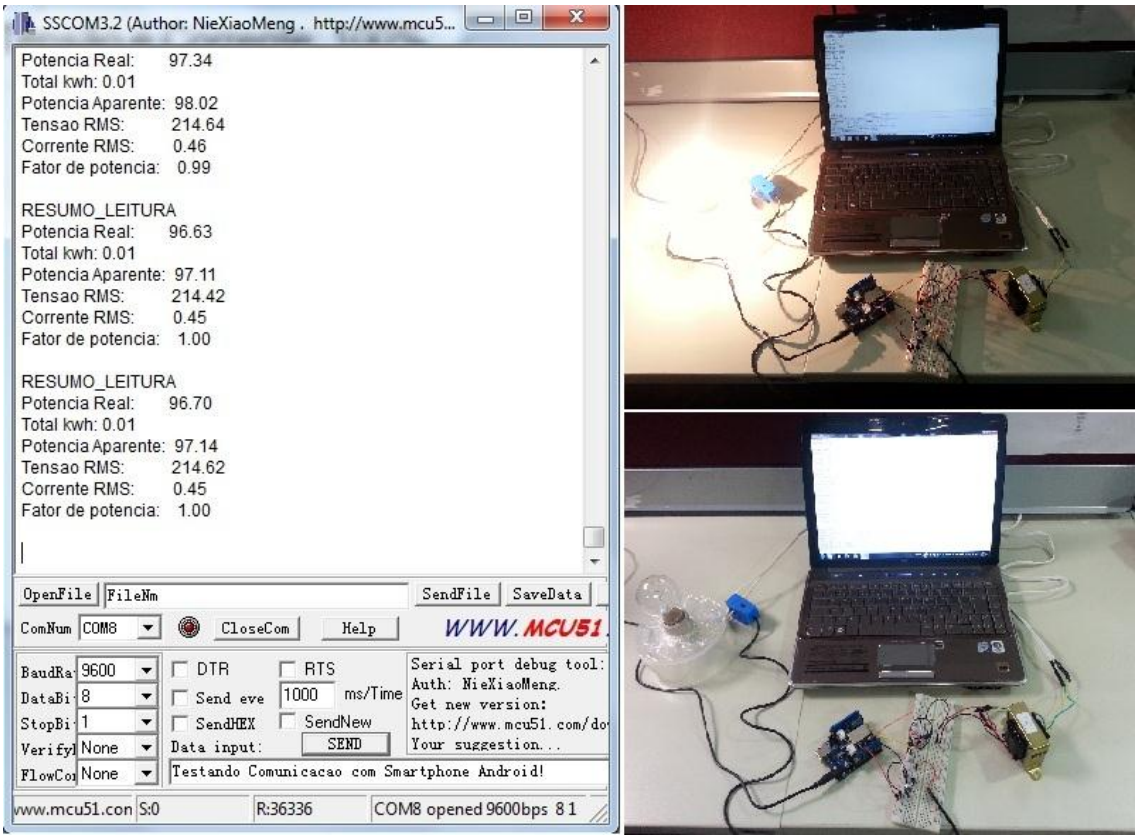

Figura 3.11 – Leitura das informações fornecidas ao ligar uma lâmpada de 100W. (Fonte: Autor)

Após concluir a fase de leitura e impressão dos dados colhidos foi possível comparar os dados teóricos com os dados advindos do protótipo. Os resultados obtidos foram: Potência real, total kWh consumidos, potência aparente, tensão RMS, corrente RMS e fator de potência. Os dados foram satisfatórios e de acordo com os cálculos esperados previamente realizados para a lâmpada utilizada como base.

3.2.3 – Gravação e listagem das medições.

Com a conclusão da montagem, testes e validação dos dados obtidos pelo protótipo, foi criado um programa Web JSP para iniciar a comunicação entre o medidor e o servidor, realizar as leituras e armazená-las e disponibilizar na interface Web os registros armazenados. A máquina utilizada como servidora possui sistema operacional Windows 7 e arquitetura 64 bits. Nela, foi criado para armazenamento de dados, no MySQL Workbench 5.2, o banco de dados DB_CONSUMO e instalado o Software Apache Tomcat para implantação do sistema.

As partes principais dos códigos de criação do banco utilizado, tabela, e partes principais do projeto criado encontram-se no apêndice.

O programa criado, implantado no próprio notebook sobre o servidor Apache Tomcat, assim que iniciado e acessado, estabelece conexão com o *shield Bluetooth* conectado ao Arduino, realiza o pareamento e inicia a gravação de dados.

Para gravação dos dados advindos do medidor, foi utilizada a biblioteca Java RXTX, que permitiu o estabelecimento da conexão *Bluetooth* do lado servidor e o recebimento de informações. A figura 3.12 mostra a parte principal do código implementado com definições de taxa de velocidade, DataBits e paridade.

```
1   CommPort commPort = portIdentifier.open(this.getClass().getName(),2000);
2
3           if ( commPort instanceof SerialPort ){
4               SerialPort serialPort = (SerialPort) commPort;
5   //          Ajuste de configuracoes de taxa de velocidade, Databits, StopBits e paridade.
6               serialPort.setSerialPortParams(9600,SerialPort.DATABITS_8,
7                                                   SerialPort.STOPBITS_1,SerialPort.PARITY_NONE);
8
9               ...
10              ..
11              if (!dadoRecebido.isEmpty()) {
12
13                          String[] dados = dadoRecebido.split("\r\n");
14
15                          if (dados.length == 6) {
16                              Consumo consumo = new Consumo();
17
18                                  consumo.setPotenciaAtiva(dados[0]);
19                                  consumo.setPotenciaAparente(dados[1]);
20                                  consumo.setTensao(dados[2]);
21                                  consumo.setCorrente(dados[3]);
22                                  consumo.setFatorPotencia(dados[4]);
23                                  consumo.setKwh(dados[5]);
24
25                                  ConsumoPersistencia consumoPersistencia = new ConsumoPersistencia();
26                                  consumoPersistencia.adiciona(consumo);
27                          }
```

Figura 3.12 – Parte principal do código de recebimento e tratamento das informações. (Fonte: Autor)

No código da figura 3.12 foi criado um vetor para recebimento dos dados do Buffer da porta COM e a instanciação do objeto consumo para gravação em banco através de uma camada de persistência. Nesta etapa, cada item do vetor corresponde a um dado da leitura realizada e esta é feita através da execução de um Thread Java e um loop que analisa repetidamente a chegada de dados. Para fins experimentais, a leitura e gravação foram realizadas de 1 em 1 segundo, para que o sistema e a gravação fossem testados de forma intensiva.

Simultaneamente é possível listar os registros e selecionar períodos de consulta sem interferir na gravação dos dados por se tratarem de processos diferentes.

Para visualização dos dados, foi utilizado o *plugin* do jQuery, *Datatables*. Este *plugin* permite que ao abrir a página sejam mostrados todos os registros armazenados em banco separados por paginação e é possível ao usuário selecionar um período de consulta, usar caixa de busca ou ordenar as colunas dos dados por ordem crescente ou decrescente. Para o propósito, como o consumo é avaliado por períodos, os registros são agrupados por dia.

No capítulo 4 são demonstrados os resultados obtidos após recebimento e gravação dos dados.

CAPÍTULO 4 – TESTES E RESULTADOS DA APLICAÇÃO DO MODELO

O modelo apresentou leituras satisfatórias de medição, assim como as transmitiu de forma eficiente e sem perda tanto para o servidor como para o *Smartphone* utilizado para testes de comunicação *Bluetooth*. Como demonstrado no desenvolvimento do capítulo 3, as leituras de tensão e correntes foram extremamente próximas do estimado matematicamente e das medições realizadas pelo multímetro durante os testes realizados. Como os dados são fornecidos ao usuário de forma transparente, na maior parte dos testes foram utilizados o programa SSCOM e o próprio console do programa Eclipse para visualização das informações obtidas. Para maiores detalhamentos, segue sequência de testes demonstrando a leitura dos dados com equipamentos de diferentes potências.

4.1 – Teste de medição com lâmpada de 100W.

No primeiro teste foi realizada a leitura de uma lâmpada de 100W. O sistema foi ligado e em seguida a lâmpada foi acionada. Conforme visualizado na figura 4.1, foram medidos Tensão RMS, Corrente RMS, Potência Aparente e Potência Real, Fator de Potência, e total de kWh que era incrementado com base no tempo de medição.

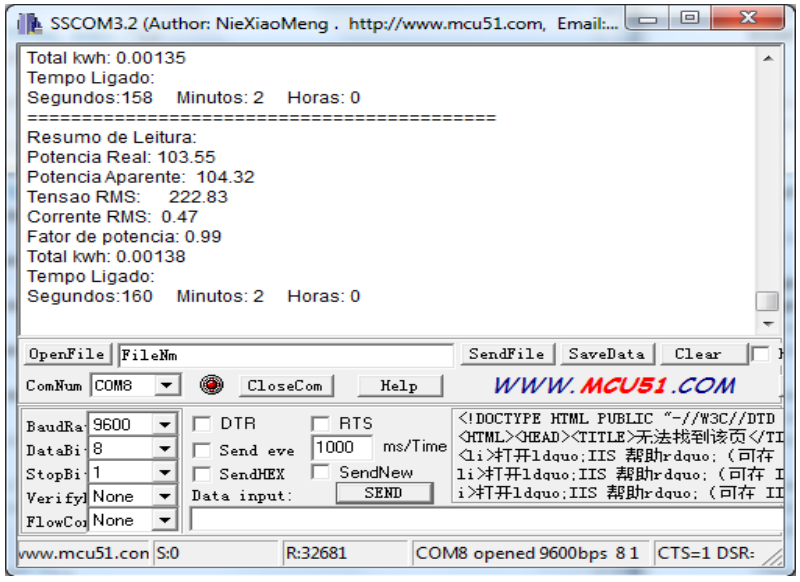

Figura 4.1 – Leitura de informações com lâmpada incandescente de 100W. (Fonte: Autor)

4.2 – Teste de medição com lâmpada de 60W.

O segundo teste foi semelhante ao primeiro, porém no lugar da lâmpada de 100W foi utilizada uma lâmpada de 60W.

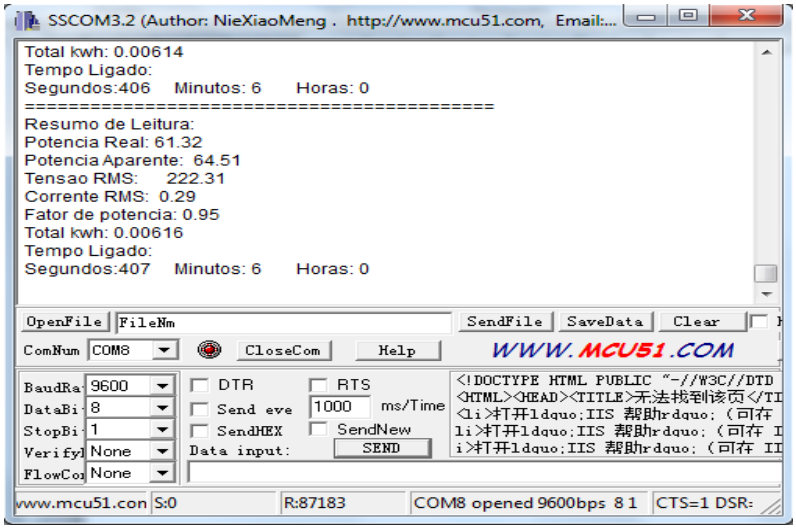

Figura 4.2 – Leitura de informações com lâmpada incandescente de 60W.
(Fonte: Autor)

4.3 – Teste de medição com ferro de passar roupa de 1200W.

No último teste foi utilizado um equipamento de maior potência no lugar das lâmpadas. O item escolhido foi um ferro de passar roupa de 1200W.

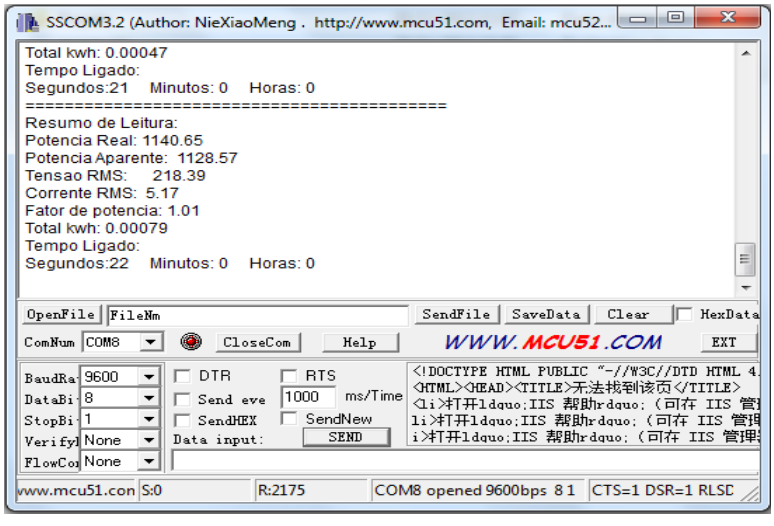

Figura 4.3 – Leitura de informações com ferro de passar roupa de 1200W.
(Fonte: Autor)

Os valores de potência colhidos sofrem alguma variação se comparados às potências nominais declaradas nas informações dos produtos utilizados. Esta diferença é esperada visto que os valores declarados são números aproximados e utilizados comercialmente.

4.4 – Visualização dos dados gravados em banco de dados

Na figura 4.4 é mostrada a tabela do banco de dados onde os dados são armazenados após passarem pelo servidor, serem tratados e armazenados.

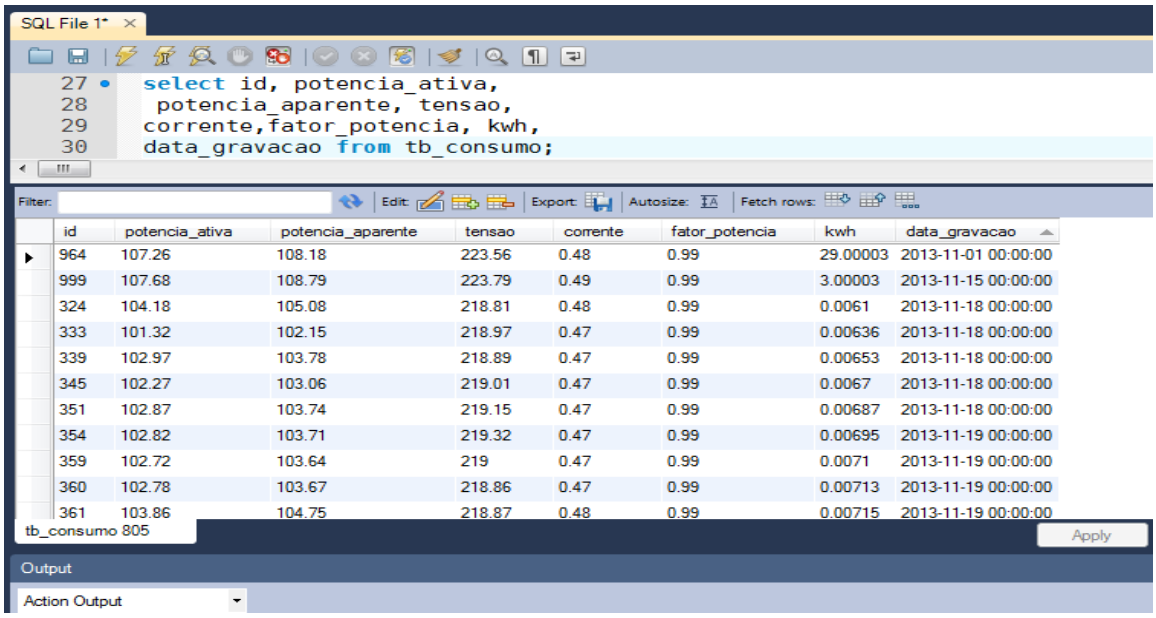

Figura 4.4 – Listagem na tabela tb_consumo criada para armazenamento das consultas. (Fonte: Autor)

Na ilustração é possível perceber que as informações foram colhidas e enviadas ao servidor com êxito após ser realizada uma consulta na própria base de dados. Pelo comando "*select*" foram consultados os itens principais como: potência ativa, potência aparente, tensão, corrente, fator de potencia, kwh e data de gravação de cada leitura realizada e armazenada na tabela "tb_consumo". A partir desta tabela, a aplicação Web é alimentada e as informações são disponibilizadas via browser ao usuário como demonstrado no item 4.5.

4.5 – Visualização da Interface Web

Na visão do cliente, a interface Web disponibiliza a visualização do total de kWh consumidos e o valor em R$(Reais) acumulados equivalentes ao total de kWh por dia. Na figura 4.5 é mostrada a interface Web implementada em JSP e Datatables.

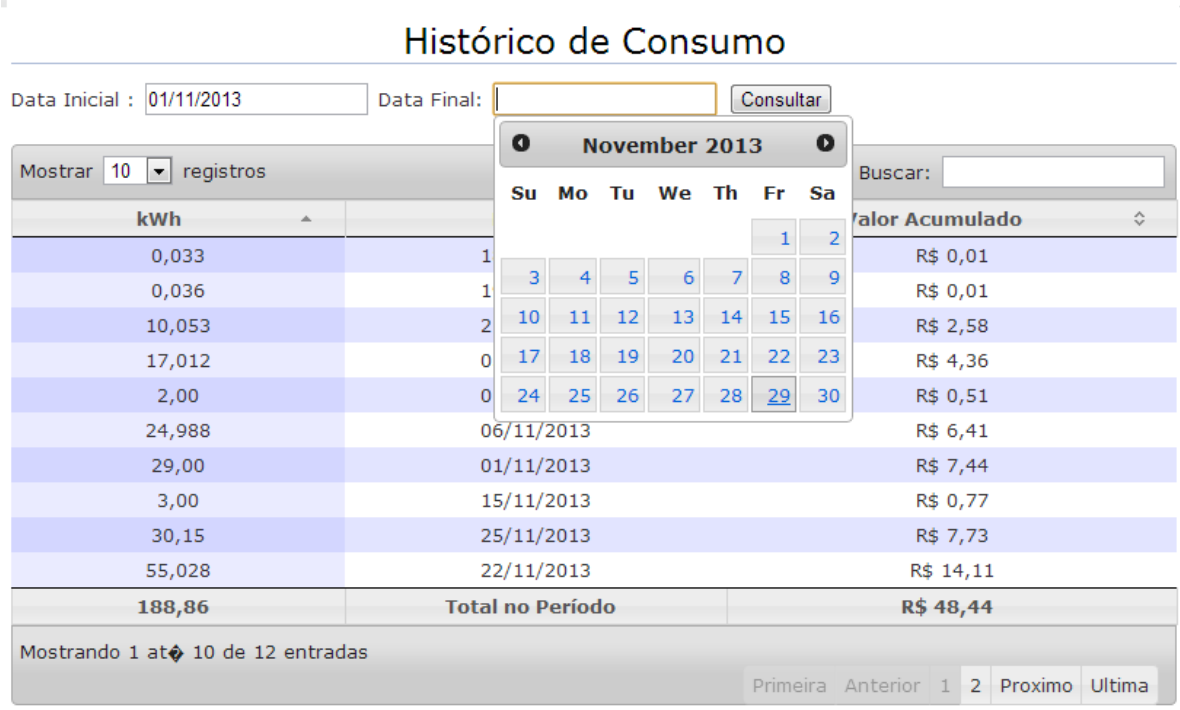

Figura 4.5 – Listagem de registros via interface Web.
(Fonte: Autor)

A interface Web construída permite como previsto a filtragem de períodos personalizados de acordo com a preferência do usuário e pode ser ordenada por kWh, Data e Valor Acumulado de forma crescente e decrescente.

CAPÍTULO 5 – APLICAÇÃO PRÁTICA DO MODELO PROPOSTO

5.1 - Apresentação da área de Aplicação do Modelo

Para o caso atual o modelo é aplicado em uma residência onde se conhece o histórico de consumo e a demanda média de energia. O conhecimento destes fatores auxilia na aplicação do modelo visto que servem de parâmetro para comparação entre os dados já obtidos anteriormente e os dados colhidos com a implementação do modelo. Considerando que há relativa estabilidade na demanda de energia, ou seja, não há variações extremas de consumo entre os períodos de medição, a comparação servirá como base para analisar os resultados de medição obtidos com o projeto.

O modelo proposto pode ser aplicado a residências em que se deseja monitorar o consumo de energia e até mesmo em outros ambientes que forneçam a infraestrutura necessária para sua implementação desde que observadas as restrições de distância delimitadas neste protótipo.

5.2 - Descrição da Aplicação do Modelo

O modelo construído permite o monitoramento do consumo de energia de forma não invasiva. O conjunto de hardware desenvolvido se apresenta portável devido ao pequeno porte e de fácil integração ao sistema residencial. O sistema necessita para seu funcionamento de uma fonte de energia de 12V para o conjunto Arduino e *Shield Bluetooth* e também de um ponto de energia para ligação do transformador que fornecerá proporcionalmente as informações de tensão para o circuito.

Para o funcionamento da solução, o módulo hardware é ligado e aguarda requisição de pareamento do programa Web. Assim que o usuário acessa, via Browser, a aplicação “GESTAO_ENERGIA_WEB” implantada na máquina servidor, a requisição de pareamento é enviada e a comunicação estabelecida. Desta forma, as informações obtidas no módulo de hardware são enviadas ao servidor de forma transparente ao usuário. Para fins de teste as

gravações ocorreram em intervalos de segundos possibilitando o incremento do consumo na tela de listagem. Para funcionamento real podem ser parametrizados via código Java o intervalo de tempo em que ocorrerá transferência de dados. No caso de residência, por exemplo, podem ser parametrizados leituras 4 vezes ao dia, de 6 em 6 horas, iniciando às 00:00 horas e possibilitando a disponibilidade de consumo ao final de cada período do dia, sendo, madrugada, manhã, tarde e noite ou ainda em intervalos personalizados de acordo com a necessidade e desejo do usuário.

5.2.1 - Dificuldades Encontradas

Na leitura dos dados ocorreram interferências externas ao sistema. Observou-se que o notebook quando alimentado por fonte externa, causava interferências no sistema se os mesmos estivessem próximos.

Quanto a codificação, houve dificuldade para captar os dados fornecidos pelo Arduino através do sistema Java utilizando a biblioteca RXTX.COM. No momento da captura, algumas informações que se encontravam no *Buffer* acabavam despadronizando o que se esperava de chegada no lado servidor. Para a solução do problema foi regulado o tamanho do *Buffer* no código da biblioteca para que os dados não chegassem destorcidos estabelecendo um tempo de *sleep* na *Thread* de leitura da aplicação de modo que aguardasse a chegada dos dados fornecidos pelo protótipo.

5.3 – Custos do modelo proposto

Na figura 5.1 é mostrada a relação de componentes utilizados no projeto e seus respectivos custos. Ao final é mostrado o custo total para implementação do projeto.

Componente	Preço
Microcontrolador Arduino UNO	R$ 71,67
Cabo USB	R$ 5,00
Filtro de Linha	R$ 19,90
Shield Bluetooth	R$ 74,89
Sensor de Corrente não invasivo	R$ 45,57
Protoboard 830 Furos Minipa MP-830A	R$ 13,00
Fonte de Alimentação 12V/ 1A	R$ 12,90
Resistores, Fios Jumper, Régua, Lâmpadas e outros.	R$ 45,00
Transformador Abaixador de Tensão	R$ 16,90
Caixa para colocação dos componentes	R$ 15,00
Frete de importação de componentes	R$ 100,00
Total	**R$ 419,83**

Figura 5.1 – Custos do Projeto.
(Fonte: Autor)

5.4 - Avaliação Global do Modelo

Quanto à implementação e instalação, o modelo é considerado de fácil instalação devido a prática introdução do sensor de corrente em qualquer circuito.

O modelo mostrou-se eficiente na medição de dados, assim como na demonstração dos resultados para o usuário. Possibilitou o estabelecimento de uma conexão estável e confiável e correta demonstração via aplicações de software.

Apesar da limitação referente ao tipo do *Bluetooth* utilizado visto que não permite que o servidor seja posicionado por cerca de mais de 10 metros de onde se encontra o protótipo de hardware, o conjunto funcionou bem dentro do raio de distância delimitado.

CAPÍTULO 6 - CONCLUSÃO

6.1 – Conclusões

Neste projeto foi construído um gerenciador de consumo de energia residencial composto de dois módulos distintos. Sendo eles: um módulo de hardware e um módulo de software.

Quanto ao módulo de hardware foi implementado um protótipo de medidor de energia utilizando como dispositivo centralizador o micro controlador Arduino UNO R3. Ao micro controlador foi adicionado um algoritmo para captação e recebimento dos dados do sensor de corrente e do transformador de tensão e posterior envio por meio de tecnologia sem fio para um servidor.

Para transmissão até o servidor, o *Shield Bluetooth* utilizado, acoplado ao Arduino, trabalhou da forma esperada enviando os dados colhidos sem prejudicar ou alterar a informação e sem sofrer interferência de outras frequências tanto nos testes de comunicação quanto nas transmissões das medições. Além disso, apresentou fácil e rápido pareamento com outros dispositivos com suporte a *Bluetooth* como mostrado no desenvolvimento projeto. Em contrapartida, não pode ser utilizado para pareamentos a distâncias maiores do que 10 metros por se tratar de dispositivo com tecnologia *Bluetooth* Classe 2. Considerando o custo-benefício do projeto, o *Shield* atendeu às necessidades do escopo do projeto.

O primeiro módulo pode ser embarcado ao medidor de energia de forma não invasiva em decorrência da praticidade e facilidade de manuseio do sensor de corrente SCT013 utilizado. O sensor de corrente envolve o fio da fase do circuito a ser monitorado sem interferir em absolutamente nada na estrutura residencial, permitindo praticidade e segurança na instalação e necessitando apenas de fonte de alimentação de 12V e um ponto de energia para ligação do transformador de energia.

Quanto ao módulo de software foi implementada uma aplicação Java Web, capaz de receber via *Bluetooth* os dados provindos do micro controlador, armazena-los em uma base de dados e posteriormente lista-los de forma simplificada e prática para o usuário de energia. Houve dificuldade em tratar os dados recebidos do primeiro módulo dado que as informações chegavam via buffer da porta COM, mas esta questão foi resolvida estabelecendo padrões para recebimento dos dados e estabelecimento de intervalos de comunicação e leitura entre os módulos.

O gerenciador cumpriu seu propósito visto que a medição de corrente, tensão, potência ativa e aparente e kWh consumidos foram mensurados e monitorados com sucesso, tanto nos testes isolados quanto nos testes do conjunto, assim como gravados e disponibilizados. Foi possível obter também uma estimativa em reais (R$) do consumo definindo a taxa de kWh com base nos dados da ANEEL e acompanhamento do consumo.

Desta forma, o projeto cumpriu tanto o objetivo geral quanto os objetivos específicos, possibilitando ao usuário não só a manter um histórico de consumo de energia por dia e por períodos personalizados acessando uma simples interface Web como monitorar o consumo a fim de gerir, economizar e monitorar possíveis desperdícios.

Além de o projeto contribuir para a aproximação do usuário com os seus gastos e consequentemente na conscientização do uso de energia, possibilita a comodidade de mensurar sem esforço o valor da conta no intervalo requerido e o auxílio na identificação de problemas elétricos na rede elétrica da casa ou de equipamentos com defeito que estejam interferindo indevidamente no consumo de energia.

O gerenciador no geral apresentou bom custo-benefício visto que atendeu as expectativas permitindo um modo inovador no acompanhamento do consumo de energia sem demandar grandes recursos monetários.

6.2 - Sugestões para Trabalhos Futuros

Como sugestão de trabalhos futuros e aprimoramento do projeto em questão, podem ser citados alguns aspectos referentes à comunicação, alimentação do dispositivo e melhorias no circuito elétrico.

Quanto à comunicação sem fio entre o dispositivo montado e o notebook servidor, podem ser utilizados outros tipos de comunicação sem fio ou até mesmo comunicação *Bluetooth* da Classe 1 para que o dispositivo possa ficar a uma distância de até 100 metros do servidor. Essa alteração possibilitaria maior mobilidade ao servidor garantindo que a conexão pudesse ser estabelecida de uma distância maior.

Sobre à alimentação do dispositivo, pode ser utilizado uma bateria em substituição a fonte de alimentação utilizada para alimentação do conjunto Arduino e *Shield Bluetooth*. Essa alteração permitiria maior mobilidade e portabilidade do dispositivo e independência de um ponto de energia para alimentação.

Quanto ao circuito elétrico, pode ser desenvolvido um circuito impresso com a utilização de resistores com tolerância menor a fim de obter melhor precisão nos resultados.

REFERÊNCIAS

ALEXANDER, SADIKU. **Fundamentos de Circuitos Elétricos.** São Paulo: Bookman, 2003;

ARDUINO. **Examples.** Disponível em: <www.arduino.cc/en/Tutorial/HomePage>. Acessado em: Outubro de 2013.

BIBLIOTECA BLUETOOTH SHIELD DEMO CODE, **SEEDSTUDIO.** Disponível em <http://www.seeedstudio.com/wiki/images/3/30/BluetoothShieldDemoCode.zip>. Acessado em Outubro de 2013.

BIBLIOTECA NEW SOFTWARE SERIAL, **SEEDSTUDIO.** Disponível em <http://arduiniana.org/NewSoftSerial/NewSoftSerial10c.zip>. Acessado em Outubro de 2013.

BIBLIOTECA RXTX, **RXTX.QBANG.ORG.** Disponível em http://rxtx.qbang.org/wiki/index.php/ >. Acessado em Outubro de 2013.

DIÁRIO DA MANHÃ. **Brasil desperdiça, por ano, 10% de energia produzida.** Disponível em <http://www.dm.com.br/texto/85003>. Acessado em Outubro de 2013.

EMPRESA DE PESQUISA ENERGÉTICA. **Resenha Mensal do Mercado de Energia Elétrica.** Disponível em <www.epe.gov.br/ResenhaMensal/20131030_1.pdf>. Acessado em Outubro de 2013.

FOROUZAN, Behrouz A. **Comunicação de dados e redes de computadores.** 3ª edição;

HINRICHS, Roger A.; KLEINBACH, Merlin; REIS, Lineu Belico dos. **Energia e Meio Ambiente**. Tradução da 4ª edição norte-americana;

NILSSON, RIEDEL. **Circuitos Elétricos.** 8ª edição. São Paulo: Pearson, 2009;

OPERADOR NACIONAL DO SISTEMA ELÉTRICO. Disponível em <http://www.ons.org.br/home/>. Acessado em Outubro de 2013.

ROBERTS, Michael. **Arduino Básico.** 1ª edição.Novatec,2012 ;

SENSOR DE CORRENTE NÃO INVASIVO, **SEEDSTUDIO.** Disponível em <http://www.seeedstudio.com/depot/images/product/2009511142810295.gif>. Acessado em Outubro de 2013.

SOLON, M.F.; **Medição de Energia Elétrica**. 4ª edição. Rio de Janeiro, Brasil. LTC.

TANENBAUM, A. S. J.; WETHERALL, D. **Redes de Computadores**. 5ª. edição. São Paulo: Pearson.

TIPLER, Paul A.; MOSCA, Gene. **Física: Eletricidade e Magnetismo, Ótica**. 5ª edição. Volume 2, Rio de Janeiro, Brasil. LTC;

APÊNDICE

Código 1 – Código de leitura de corrente, tensão e potências utilizado nos testes de comunicação.

```
// EmonLibrary openenergymonitor.org, Licence GNU GPL V3

//Includes referentes a comunicacao bluetooth
#include <SoftwareSerial.h>            //Software Serial Port
#define RxD 6
#define TxD 7

#define DEBUG_ENABLED  1

SoftwareSerial blueToothSerial(RxD,TxD);

//Includes referentes a biblioteca Emonlib
#include "EmonLib.h"                   // Include Emon Library
EnergyMonitor emon1;                   // Create an instance


  double total_KWH_acumulado = 0;
  double valor_consumido_acumulado = 0;
  double potencia_ativa = 0;
  double potencia_aparente = 0;
  double tensao = 0;
  double corrente = 0;
  double fator_de_potencia = 0;
  unsigned long tempo_medicao = 0;


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

  //definicao de entrada e saida bluetooth.
  pinMode(RxD, INPUT);
  pinMode(TxD, OUTPUT);

  //definicao de entrada e saida  para botao e comunicacao
  pinMode(13, OUTPUT);
  pinMode(10, INPUT);
  digitalWrite(10, 1);

   //dados de calibracao
  emon1.voltage(3, 255.26, 1.7);  // Voltage: input pin, calibration,
phase_shift
  emon1.current(2, 60.6);       // Current: input pin, calibration.

  //seta configuracoes da conexao bluetooth
  setupBlueToothConnection();
  //blueToothSerial.println("=======TESTE DE MEDIÇÃO DE TENSÃO=======");
}

void loop()
{
        char recvChar;

    while(1){

       // Calculate all. No.of wavelengths, time-out
```

```
      emon1.calcVI(20,2000);

      Serial.read();

      potencia_ativa = emon1.realPower;
      tensao = emon1.Vrms;

      if(potencia_ativa > 0 && tensao < 250){

         potencia_aparente = emon1.apparentPower;
         corrente = emon1.Irms;
         fator_de_potencia = emon1.powerFactor;

        //potencia_ativa eh convertida para kwh e atribuida a
total_KWH_acumulado.
         total_KWH_acumulado = total_KWH_acumulado +
(potencia_ativa/3600000);
      }

          Serial.flush();

blueToothSerial.println("=========================================");
          blueToothSerial.println("Resumo de Leitura: ");
          blueToothSerial.print("Potencia Real: ");
          blueToothSerial.println(potencia_ativa);

          blueToothSerial.print("Potencia Aparente:  ");
          blueToothSerial.println(potencia_aparente);

          blueToothSerial.print("Tensao RMS:      ");
          blueToothSerial.println(tensao);

          blueToothSerial.print("Corrente RMS:  ");
          blueToothSerial.println(corrente);

          blueToothSerial.print("Fator de potencia: ");
          blueToothSerial.println(emon1.powerFactor);

          blueToothSerial.print("Total kwh: ");
          blueToothSerial.println(total_KWH_acumulado,5);

          blueToothSerial.println("Tempo Ligado:         ");

          tempo_medicao = millis();
          tempo_medicao = tempo_medicao/1000;
          blueToothSerial.print("Segundos:");
          blueToothSerial.print(tempo_medicao);



        delay(1000);

    }
}
//seta configuracoes da conexao bluetooth
void setupBlueToothConnection()
{
  blueToothSerial.begin(38400); //Set BluetoothBee BaudRate to default baud
rate 38400
  blueToothSerial.print("\r\n+STWMOD=0\r\n"); //set the bluetooth work in
slave mode
```

```
  blueToothSerial.print("\r\n+STNA=Shield_Nery\r\n"); //set the bluetooth
name as "SeeedBTSlave"
  blueToothSerial.print("\r\n+STOAUT=1\r\n"); // Permit Paired device to
connect me
  blueToothSerial.print("\r\n+STAUTO=0\r\n"); // Auto-connection should be
forbidden here
  delay(2000); // This delay is required.
  blueToothSerial.print("\r\n+INQ=1\r\n"); //make the slave bluetooth
inquirable
  Serial.println("O Shield esta pronto!");
  delay(2000); // This delay is required.
  blueToothSerial.flush();
}
```

Código 2 – Código de leitura de corrente, tensão e potências.

```
// EmonLibrary openenergymonitor.org, Licence GNU GPL V3

//Includes referentes a comunicacao bluetooth
#include <SoftwareSerial.h>           //Software Serial Port
#define RxD 6
#define TxD 7

#define DEBUG_ENABLED  1

SoftwareSerial blueToothSerial(RxD,TxD);

//Includes referentes a biblioteca Emonlib
#include "EmonLib.h"                   // Include Emon Library
EnergyMonitor emon1;                   // Create an instance

  float potencia_ativa_acumulada = 0;
  float valor_acumulado = 0;

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

  //definicao de entrada e saida bluetooth.
  pinMode(RxD, INPUT);
  pinMode(TxD, OUTPUT);

  //definicao de entrada e saida  para botao e comunicacao
  pinMode(13, OUTPUT);
  pinMode(10, INPUT);
  digitalWrite(10, 1);

   //atualizar dados de calibracao
  emon1.voltage(3, 255.26, 1.7);  // Voltage: input pin, calibration, phase_shift
  emon1.current(2, 60.6);       // Current: input pin, calibration.

  //seta configuracoes da conexao bluetooth
  setupBlueToothConnection();

}

void loop()
{

    char recvChar;

    while(1){


      int leitura = digitalRead(10);
```

```
    if(leitura == 0){
      blueToothSerial.println("Botao pressionado");
      delay(1000);
    }

  //DISPOSITIVO PAREADO PARA SHIELD.
  if(blueToothSerial.available()){
    recvChar = blueToothSerial.read();
    Serial.print(recvChar);
  }

    // Calculate all. No.of wavelengths, time-out
    emon1.calcVI(20,2000);


    float potencia_ativa = 0;

    potencia_ativa = emon1.realPower;

    Serial.read();

    blueToothSerial.println("RESUMO_LEITURA");

    blueToothSerial.print("Potencia Real:        ");
    blueToothSerial.println(potencia_ativa);


    //potencia_ativa eh convertida para kwh e atribuida a potencia_ativa_acumulada.
    potencia_ativa_acumulada = potencia_ativa_acumulada + (potencia_ativa/3600000);

    blueToothSerial.print("Total kwh:         .");
    blueToothSerial.println(potencia_ativa_acumulada);

    blueToothSerial.print("Potencia Aparente:  ");
    blueToothSerial.println(emon1.apparentPower);

    blueToothSerial.print("Tensao RMS:          ");
    blueToothSerial.println(emon1.Vrms);

    blueToothSerial.print("Corrente RMS:        ");
    blueToothSerial.println(emon1.Irms);

    blueToothSerial.print("Fator de potencia:    ");
    blueToothSerial.println(emon1.powerFactor);


    blueToothSerial.println("");

  }
}
```

```
//seta configuracoes da conexao bluetooth
void setupBlueToothConnection()
{
  blueToothSerial.begin(38400); //Set BluetoothBee BaudRate to default baud rate 38400
  blueToothSerial.print("\r\n+STWMOD=0\r\n"); //set the bluetooth work in slave mode
  blueToothSerial.print("\r\n+STNA=Shield_Nery\r\n"); //set the bluetooth name as
"SeeedBTSlave"
  blueToothSerial.print("\r\n+STOAUT=1\r\n"); // Permit Paired device to connect me
  blueToothSerial.print("\r\n+STAUTO=0\r\n"); // Auto-connection should be forbidden here
  delay(2000); // This delay is required.
  blueToothSerial.print("\r\n+INQ=1\r\n"); //make the slave bluetooth inquirable
  Serial.println("O Shield esta pronto!");
  delay(2000); // This delay is required.
  blueToothSerial.flush();
}
```

Código 3 – Código de leitura de corrente, tensão e potências com somente números para envio ao servidor.

```
// EmonLibrary openenergymonitor.org, Licence GNU GPL V3

//Includes referentes a comunicacao bluetooth
#include <SoftwareSerial.h>            //Software Serial Port
#define RxD 6
#define TxD 7

#define DEBUG_ENABLED  1

SoftwareSerial blueToothSerial(RxD,TxD);

//Includes referentes a biblioteca Emonlib
#include "EmonLib.h"                   // Include Emon Library
EnergyMonitor emon1;                   // Create an instance

  double KWH_consumido = 0;
  double potencia_ativa = 0;
  double potencia_aparente = 0;
  double tensao = 0;
  double corrente = 0;
  double fator_de_potencia = 0;
  unsigned long tempo_medicao = 0;


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

  //definicao de entrada e saida bluetooth.
  pinMode(RxD, INPUT);
  pinMode(TxD, OUTPUT);

  //definicao de entrada e saida  para botao e comunicacao
  pinMode(13, OUTPUT);
  pinMode(10, INPUT);
  digitalWrite(10, 1);

  // configuracao de calibracao
  emon1.voltage(3, 255.26, 1.7);  // Voltage: input pin, calibration,
phase_shift
  emon1.current(2, 60.6);       // Current: input pin, calibration.

  //seta configuracoes da conexao bluetooth
  setupBlueToothConnection();

}

void loop()
{

    char recvChar;

    while(1){

       // Calculate all. No.of wavelengths, time-out
      emon1.calcVI(20,2000);

      potencia_ativa = emon1.realPower;
```

```
      tensao = emon1.Vrms;

      if(potencia_ativa > 0 && tensao < 250){

         potencia_aparente = emon1.apparentPower;
         corrente = emon1.Irms;
         fator_de_potencia = emon1.powerFactor;

         KWH_consumido = potencia_ativa/3600000;
      }

          Serial.flush();

          blueToothSerial.println(potencia_ativa);

          blueToothSerial.println(potencia_aparente);

          blueToothSerial.println(tensao);

          blueToothSerial.println(corrente);

          blueToothSerial.println(emon1.powerFactor);

          blueToothSerial.println(KWH_consumido,6);


          tempo_medicao = millis();
          tempo_medicao = tempo_medicao/1000;

      delay(1000);

    }
}


//seta configuracoes da conexao bluetooth
void setupBlueToothConnection()
{
  blueToothSerial.begin(38400); //Set BluetoothBee BaudRate to default baud
rate 38400
  blueToothSerial.print("\r\n+STWMOD=0\r\n"); //set the bluetooth work in
slave mode
  blueToothSerial.print("\r\n+STNA=Shield_Nery\r\n"); //set the bluetooth
name as "SeeedBTSlave"
  blueToothSerial.print("\r\n+STOAUT=1\r\n"); // Permit Paired device to
connect me
  blueToothSerial.print("\r\n+STAUTO=0\r\n"); // Auto-connection should be
forbidden here
  delay(2000); // This delay is required.
  blueToothSerial.print("\r\n+INQ=1\r\n"); //make the slave bluetooth
inquirable
  Serial.println("O Shield esta pronto!");
  delay(2000); // This delay is required.
  blueToothSerial.flush();
}
```

Código 4 – Classe Java que estabelece comunicação com Shield e recebe informações

```
package br.com.nery.util;

import gnu.io.CommPort;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.sql.SQLException;

import br.com.nery.modelo.Consumo;
import br.com.nery.persistencia.ConsumoPersistencia;

public class TwoWaySerialComm
{
    public TwoWaySerialComm()
    {
        super();
    }


    public void connect ( String portName ) throws Exception
    {
        CommPortIdentifier portIdentifier =
CommPortIdentifier.getPortIdentifier(portName);
        if ( portIdentifier.isCurrentlyOwned() )
        {
            System.out.println("Error: Port is currently in use");
        }
        else
        {
            CommPort commPort =
portIdentifier.open(this.getClass().getName(),2000);

            if ( commPort instanceof SerialPort )
            {
                SerialPort serialPort = (SerialPort) commPort;

serialPort.setSerialPortParams(9600,SerialPort.DATABITS_8,SerialPort.STOPBI
TS_1,SerialPort.PARITY_NONE);

                InputStream in = serialPort.getInputStream();
                OutputStream out = serialPort.getOutputStream();


                (new Thread(new SerialReader(in))).start();
                (new Thread(new SerialWriter(out))).start();

            }
            else
            {
                System.out.println("Error: Only serial ports are handled by
this example.");
            }
        }
    }
```

```
    /** */
    public static class SerialReader implements Runnable
    {
        InputStream in;

        int conta_Estouro = 0;

        public SerialReader ( InputStream in )
        {
            this.in = in;
        }

        public void run ()
        {

            String dadoRecebido;

            byte[] buffer = new byte[1024];
            int len = -1;
            try
            {

                while ( ( len = this.in.read(buffer)) > -1 )
                {
                    Thread.sleep(1000);

                    dadoRecebido = (new String(buffer,0,len));

                    if (!dadoRecebido.isEmpty()) {

                        String[] dados = dadoRecebido.split("\r\n");

                        if (dados.length == 6) {
                            Consumo consumo = new Consumo();

                                consumo.setPotenciaAtiva(dados[0]);
                                consumo.setPotenciaAparente(dados[1]);
                                consumo.setTensao(dados[2]);
                                consumo.setCorrente(dados[3]);
                                consumo.setFatorPotencia(dados[4]);
                                consumo.setKwh(dados[5]);

                                ConsumoPersistencia consumoPersistencia =
new ConsumoPersistencia();
                                consumoPersistencia.adiciona(consumo);
                        }

                        }
                    }

                }
            }
            catch ( IOException e )
            {
                e.printStackTrace();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
```

```
            }
        }
    }

    /** */
    public static class SerialWriter implements Runnable
    {
        OutputStream out;

        public SerialWriter ( OutputStream out )
        {
            this.out = out;
        }

        public void run ()
        {
            try
            {
                int c = 0;
                while ( ( c = System.in.read()) > -1 )
                {
                    this.out.write(c);
                }
            }
            catch ( IOException e )
            {
                e.printStackTrace();
            }
        }
    }

    public void iniciaLeitura(){

        try {
             (new TwoWaySerialComm()).connect("COM8");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```