

# Medidor de Consumo Elétrico Microcontrolado

Marcelo Gomes Bona<sup>1</sup>, Alessandro André Mainardi de Oliveira<sup>1</sup>

<sup>1</sup>Sistemas de Informação – Centro Universitário Franciscano

97.010-32 – Santa Maria – RS – Brasil

marcelogomesbona@gmail.com, alessandroandre@unifra.br

**Abstract.** *This paper aims to provide an alternative to monitor and store the amount of electricity used in a business or industry. To achieve this a board Arduino Uno Ver 3 will be used with an ATmega328 microcontroller, with the function to receive data from the sensor and perform your reading. The information will be stored on a memory card and transferred to a database, generating graphs of consumption. With the results the user can view the most appropriate times to use his power.*

**Resumo.** *O presente trabalho visa oferecer uma alternativa, para monitorar e armazenar a quantidade de energia elétrica utilizada em uma empresa ou indústria. Para alcançar este objetivo será utilizada uma placa Arduino Uno Ver. 3, com um microcontrolador ATmega328, com a função de receber os dados vindos do sensor e realizar sua leitura. As informações serão armazenadas em um cartão de memória e transferidas para um banco de dados, gerando gráficos de consumo. Com os resultados obtidos o usuário pode visualizar os horários mais adequados do uso de sua energia elétrica.*

## 1. Introdução

Este trabalho tem por intenção elaborar um dispositivo capaz de fazer a leitura da quantidade de energia elétrica utilizada em uma empresa ou indústria, os dados são armazenados em um cartão de memória e transferidos para um banco de dados, responsável por gerar as informações em forma de gráficos ou planilhas.

A escolha pela execução deste trabalho realiza-se em função da possibilidade de utilizar um dispositivo com a finalidade de informar ao usuário, os períodos do dia em que mais se consome energia elétrica.

Na construção deste trabalho, será utilizada uma placa Arduino Duemilanove, fabricada na Itália [Arduino, 2013]. Juntamente com a placa o dispositivo contará com um sensor, responsável pela leitura da energia elétrica. Como razões pela escolha do Arduino, citam-se: o baixo custo; a facilidade oferecida pela plataforma para projetar sistemas, sem a necessidade de um conhecimento avançado em eletrônica digital; e por ultimo, por oferecer uma facilidade de comunicação, realizada via USB (*Universal Serial Bus*), tecnologia difundida e amplamente presente em computadores atuais.

## 2. Objetivos

Este trabalho objetiva realizar um estudo acerca do funcionamento e características pertinentes ao Arduino e à eletrônica e, com isso, desenvolver um protótipo que seja capaz de informar em quais horários do dia, ocorre o consumo de energia elétrica em uma empresa ou indústria.

### 2.1 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- ✓ Desenvolver uma solução para leitura dos dados recebidos.
- ✓ Desenvolver um protótipo que armazene as informações sobre a energia utilizada no momento da leitura.
- ✓ Definir qual aplicação deve receber os dados coletados.
- ✓ Apresentar as leituras do sensor na forma de gráficos interativos e informativos.

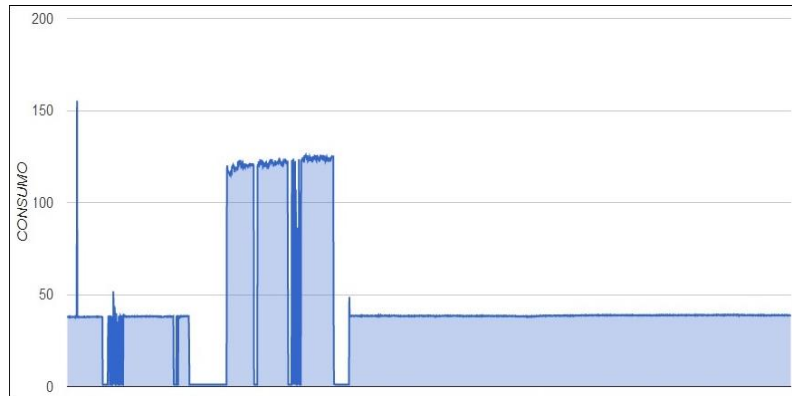
## 3. Projeto

O modelo escolhido para o projeto é o Arduino Duemilanove, utilizando-se de um microcontrolador ATmega328, onde o mesmo pode ser adaptado para diversos outros modelos existentes, esta escolha deve-se ao fato de ser um modelo básico encontrado facilmente no mercado, possuindo uma fácil conectividade a computadores convencionais. Adaptado ao Arduino está o *Shield Ethernet*, no qual é um componente que disponibiliza um *slot* para armazenar informação em um cartão de memória SD e conectividade via Rede, quando necessário [Arduino, 2013].

O conjunto destes componentes possui como principal função controlar todo o funcionamento do sistema, no qual a partir de um loop programado, pode-se ler os dados do sensor de corrente, e armazenar a informação em um cartão de memória, onde através desta leitura, é possível criar gráficos que mostram o consumo diário durante o tempo de leitura, ou pesquisar por dias individuais que é mostrado em horas.

O código fonte do Arduino, possui duas funções básicas, uma denominada `setup()`, onde são declaradas as variáveis de tempo, calibração do sensor, pinos e inicialização do cartão de memória SD, a função `loop()`, na qual a sua função principal é ler os dados do sensor, está programada para a cada segundo armazenar em uma memória externa. Para isso também foi criado um relógio, responsável pela contagem do tempo e registro de cada leitura.

Os dados coletados pelo Arduino, são gravados em um cartão de memória SD, no formato de um arquivo.sql, este deve ser importado para um servidor onde estará rodando o banco de dados pronto para receber o arquivo. Desta maneira é possível através do navegador acessar o servidor e visualizar os dados obtidos em forma de gráfico, mostrando o consumo energético em dias e horas, realizados durante o período da leitura. Conforme mostrado na Figura 1, o gráfico gerado pelo sistema.



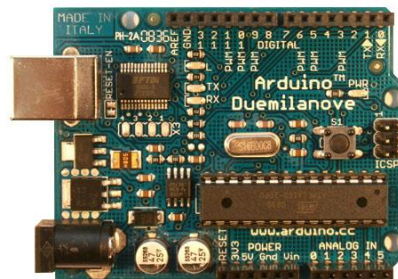
**Figura 1 – Gráfico de consumo x tempo**

## 4. Componentes

Durante a realização do projeto, foi utilizado diversos componentes, alguns eletrônicos e outros programas. Estes por sua vez, serão descritos ao longo do texto, enfatizando suas características e funções, deixando mais claro o que cada um deles deve fazer.

### 4.1 Arduino

Consiste em uma placa com microcontrolador Atmega328. Possui 14 entradas/saídas digitais, 6 entradas analógicas, um cristal oscilador de 16MHz, conexão USB, uma entrada para fonte, soquetes para ICSP, e um botão de reset. A placa contém todo o necessário para usar o micro controlador. Simplesmente conecte-a a um computador com o cabo USB ou ligue a placa com uma fonte 5 AC-DC (ou bateria). O Arduino Duemilanove seleciona automaticamente a fonte de alimentação (USB ou fonte externa). Conforme demonstrado na Figura 2 [Arduino, 2013].



**Figura 2 – Arduino Duemilanove [Arduino 2014].**

**Tabela 1 : Características Arduino Duemilanove [Arduino 2014].**

Tensão de entrada	7V ~ 12V
Tensão de saída	5V
Portas digitais I/O	14
Portas analógicas	6
Dimensões	7cm x 5,5cm

## 4.2 Shield Ethernet com slot cartão SD

O Ethernet Shield tem uma conexão RJ-45 padrão, com um transformador de linha integrado assim como Ethernet habilitado. Há um slot para cartão micro-SD a, que pode ser usado para armazenar arquivos para servir através da rede. É compatível com a Arduino Duemilanove e UNO (usando a biblioteca de Ethernet). O leitor de cartão microSD onboard é acessível através da Biblioteca SD. Conforme mostrado na Figura 3 [Arduino, 2013].



**Figura 3 – Ethernet Shield.**

**Tabela 2 : Características Ethernet Shield [Arduino 2014].**

Tensão de operação	5V
Velocidade de transmissão de dados	10/100Mb
Dimensões	7cm x 5,5cm
Tipo de cartão de memória	Micro SD
Tipo de conector de rede	RJ-45

## 4.3 Sensor de Corrente

Para a coleta da informação referente a energia elétrica, será realizada utilizando o Sensor de Corrente SCT-013-100 conforme a Figura 3 [Openenergymonitor, 2014].



**Figura 3 – Sensor de corrente.**

O dispositivo utilizado é mostrado na Figura 3, trata de um sensor que adaptado a um cabo, que conduz uma corrente elétrica, Este por sua vez, utilize-se como princípio de leitura, o efeito eletromagnético de deslocamento dos elétrons em movimento de um campo elétrico, no qual é denominado efeito *hall* [Mosser, 1995].

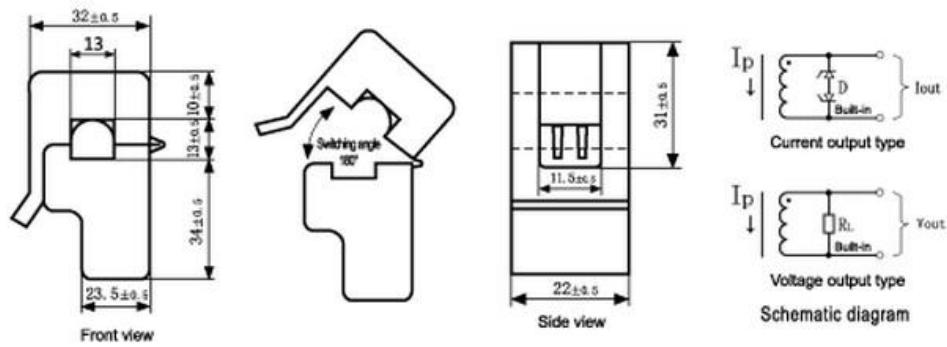


Figura 6 – Sensor adaptado a rede elétrica [Openenergymonitor, 2014].

Tabela 3 : Características do sensor SCT-013-100

Corrente de entrada	0 ~ 100A
Corrente de saída	33mA
Diâmetro do fio de abertura	13mm
Temperaturas de trabalho	-25°C ~ 70°C
Peso	63g

Através do desenho da Figura 6, é possível ver a espessura do fio para leitura, tamanho do *plug*, grau de abertura do adaptador e os pólos internos do sensor. Trata-se de um sensor não-invasivo, portanto não precisa estar conectado eletricamente ao fio, evitando assim um contato direto com a eletricidade, para obter sua medição, basta que o cabo elétrico passe por dentro do sensor, sem a necessidade de cortar fios ou descasca-los, exemplo Figura 7.

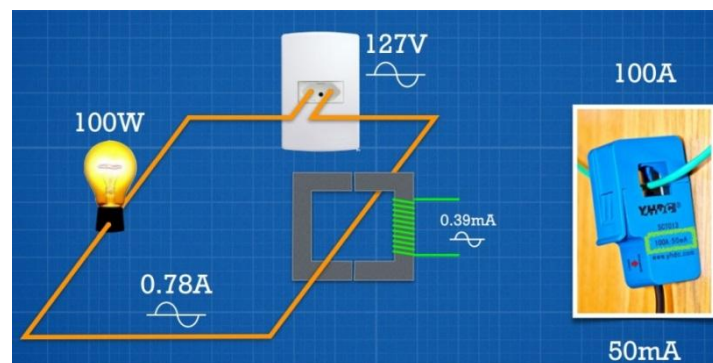


Figura 7 – Sensor adaptado a corrente elétrica [Mlemos, 2014].

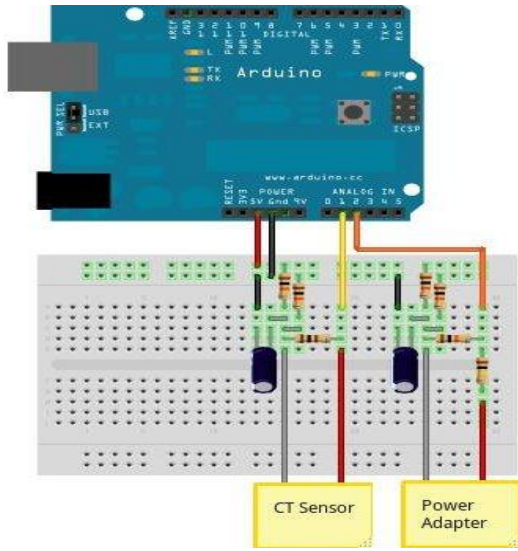
#### 4.4 Fonte Elétrica

Utilizou-se uma fonte elétrica AC Mod FTP 901 com entrada 127/220 VAC – 60Hz Saída 9v Ac Ac – 1A, que alimenta o Arduino o Shield Ethernet e o Sensor assim como todo o resto do circuito [Openenergymonitor, 2014].

## 4.5 Circuito

O sistema é composto por um circuito, onde pode ser montado em uma *protoboard*, utilizando resistores e capacitores, é possível chegar ao modelo ilustrado pela Tabela 4, onde possui especificações de potências e quantidades de cada componente [Alexander, 2003]. O Arduino utilizado no exemplo Arduino Duemilanove [Arduino, 2014].

**Tabela 4 – Componentes do circuito.**



Tipo	Quantidade	Potência
Resistor	2x	100kOhm
Resistor	4x	470kOhm
Resistor	1x	33kOhm
Capacitor	2x	10uF

## 4.6 Valores Obtidos

É demonstrado na serial do Arduino os valores recebidos pelo sensor, assim como dia da leitura, hora, minuto e segundo, foi observado uma pequena sensibilidade do sensor quando o mesmo é manuseado durante a leitura, os valores podem sofrer uma leve alteração de acordo com este movimento. É ilustrado na Figura 8, a leitura de uma lâmpada incandescente de 40w.

Escrevendo no arquivo leitura.sql...	DIA:2 0 Hs 4min 0seg	Irms: 0.32 A	Potencia: 40.48 W
Escrevendo no arquivo leitura.sql...	DIA:2 0 Hs 4min 1seg	Irms: 0.32 A	Potencia: 40.37 W
Escrevendo no arquivo leitura.sql...	DIA:2 0 Hs 4min 2seg	Irms: 0.32 A	Potencia: 40.54 W
Escrevendo no arquivo leitura.sql...	DIA:2 0 Hs 4min 3seg	Irms: 0.32 A	Potencia: 40.63 W
Escrevendo no arquivo leitura.sql...	DIA:2 0 Hs 4min 4seg	Irms: 0.32 A	Potencia: 40.47 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 0min 0seg	Irms: 0.32 A	Potencia: 40.54 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 0min 1seg	Irms: 0.32 A	Potencia: 40.47 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 0min 2seg	Irms: 0.32 A	Potencia: 40.60 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 0min 3seg	Irms: 0.32 A	Potencia: 40.53 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 0min 4seg	Irms: 0.32 A	Potencia: 40.61 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 1min 0seg	Irms: 0.32 A	Potencia: 40.39 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 1min 1seg	Irms: 0.32 A	Potencia: 40.40 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 1min 2seg	Irms: 0.32 A	Potencia: 40.64 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 1min 3seg	Irms: 0.32 A	Potencia: 40.73 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 1min 4seg	Irms: 0.33 A	Potencia: 41.30 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 2min 0seg	Irms: 0.33 A	Potencia: 41.57 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 2min 1seg	Irms: 0.33 A	Potencia: 41.29 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 2min 2seg	Irms: 0.32 A	Potencia: 41.23 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 2min 3seg	Irms: 0.33 A	Potencia: 41.57 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 2min 4seg	Irms: 0.32 A	Potencia: 41.20 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 3min 0seg	Irms: 0.32 A	Potencia: 41.13 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 3min 1seg	Irms: 0.32 A	Potencia: 41.11 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 3min 2seg	Irms: 0.32 A	Potencia: 41.16 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 3min 3seg	Irms: 0.33 A	Potencia: 41.34 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 3min 4seg	Irms: 0.33 A	Potencia: 41.41 W
Escrevendo no arquivo leitura.sql...	DIA:2 1 Hs 4min 0seg	Irms: 0.33 A	Potencia: 41.46 W

**Figura 8 – Valores Obtidos**

## 4.7 Base de dados

Nesta parte do sistema, é levado em consideração que a leitura energética já tenha ocorrida, desta forma o arquivo gerado pelo sensor necessita ser importado para dentro de um banco de dados, no qual deve já estar previamente pronto e configurado conforme o seguinte código.

```
CREATE TABLE IF NOT EXISTS `leitura` (  
  `id` int(255) NOT NULL AUTO_INCREMENT,  
  `irms` varchar(255) NOT NULL,  
  `watts` varchar(255) NOT NULL,  
  `dia` int(255) NOT NULL,  
  `hora` int(255) NOT NULL,  
  `minuto` int(255) NOT NULL,  
  `segundo` int(255) NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=2029 ;
```

Qualquer banco de dados pode ser utilizado para a coleta e produção dos dados, a criação da tabela “leitura” deve ser a mesma ilustrada na código acima, sem alterações. Caso contrário sua mudança irá influenciar na importação dos dados, e geração dos gráficos, pois a criação do arquivo.sql, está sendo realizada pela programação feita no Arduino [Date, 2003].

## 4.8 Arquivo gerado

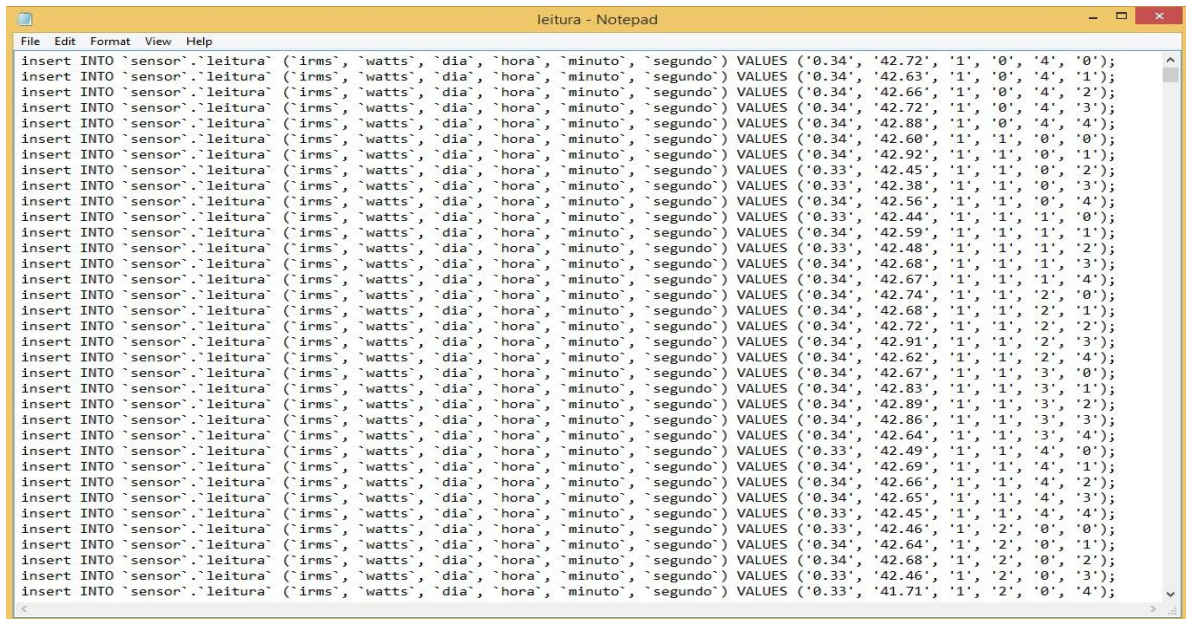
A coleta de dados realizada pelo sensor, irá gerar automaticamente um arquivo, no formato leitura.sql. Este arquivo é salvo em um Cartão de Memória, seu conteúdo, representa cada segundo de leitura, na qual é possível ver de uma forma legível na Figura 9.

Para que o Arduino faça o tratamento desta informações, e o transforme-os em dados reais, foi necessário inserir na função loop(), parâmetros de inserções no padrão SQL, tendo assim uma forma organizada de dados onde o banco de dados possa ler o arquivo, conforme mostrado no trecho de código a seguir.

```
myFile.print("insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`,  
`hora`, `minuto`, `segundo`) VALUES ('");  
  
myFile.print(Irms);  
myFile.print(", ");  
myFile.print(Potencia);  
myFile.print(", ");  
myFile.print(dia);  
myFile.print(", ");  
myFile.print(hora);  
myFile.print(", ");  
myFile.print(minuto);  
myFile.print(", ");  
myFile.print(segundo);  
myFile.print("');");  
myFile.println("");
```

O projeto utiliza o MySQL como banco de dados padrão, portanto a sintaxe do arquivo gerado está adaptada para este banco, podendo o mesmo não funcionar em

outros bancos e vir a ser necessário, uma reconfiguração do código no Arduino. Na Figura 9 podemos visualizar parte do arquivo leitura.sql pronto para ser importado.



```
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.72', '1', '0', '4', '0');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.63', '1', '0', '4', '1');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.66', '1', '0', '4', '2');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.72', '1', '0', '4', '3');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.88', '1', '0', '4', '3');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.60', '1', '1', '0', '0');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.92', '1', '1', '0', '1');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.33', '42.45', '1', '1', '0', '2');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.33', '42.38', '1', '1', '0', '3');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.56', '1', '1', '0', '4');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.33', '42.44', '1', '1', '1', '0');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.59', '1', '1', '1', '1');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.33', '42.48', '1', '1', '1', '2');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.68', '1', '1', '1', '3');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.67', '1', '1', '1', '4');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.74', '1', '1', '2', '0');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.68', '1', '1', '2', '1');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.72', '1', '1', '2', '2');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.91', '1', '1', '2', '3');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.62', '1', '1', '2', '4');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.67', '1', '1', '3', '0');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.83', '1', '1', '3', '1');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.89', '1', '1', '3', '2');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.86', '1', '1', '3', '3');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.64', '1', '1', '3', '4');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.33', '42.49', '1', '1', '4', '0');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.69', '1', '1', '4', '1');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.66', '1', '1', '4', '2');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.65', '1', '1', '4', '3');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.33', '42.45', '1', '1', '4', '4');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.33', '42.46', '1', '2', '0', '0');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.64', '1', '2', '0', '1');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.34', '42.68', '1', '2', '0', '2');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.33', '42.46', '1', '2', '0', '3');
insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`, `hora`, `minuto`, `segundo`) VALUES ('0.33', '41.71', '1', '2', '0', '4');
```

Figura 9 – Arquivo pronto para ser importado

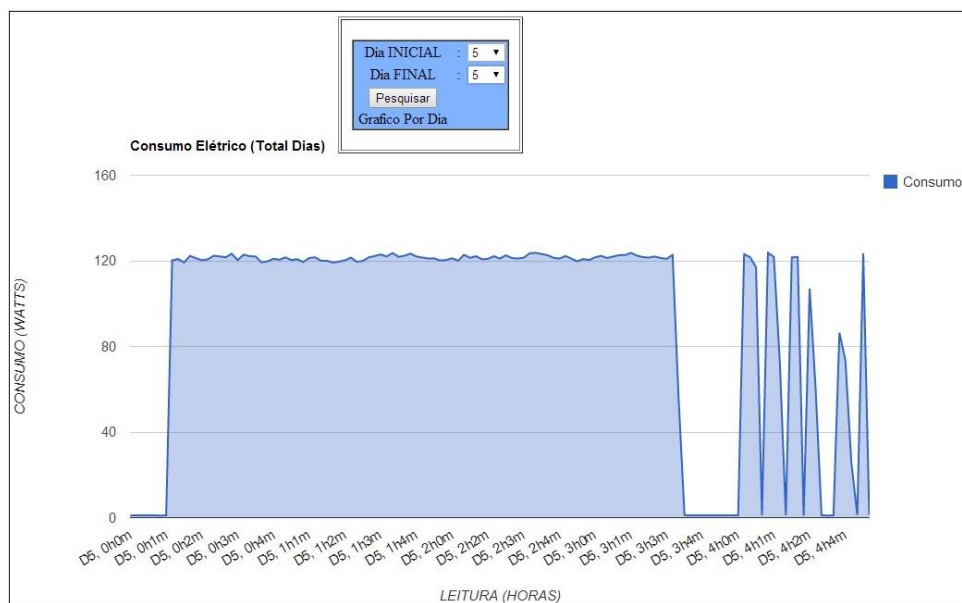
#### 4.9 Gráfico de consumo

Neste trecho, os dados de leitura contidos no arquivo já estão inseridos no banco de dados, podendo assim ser gerado seus gráficos para serem visualizados pelo cliente utilizando qualquer navegador e acessando o computador onde o banco de dados está instalado [Google Charts, 2014]. A figura 10 e Figura 11 ilustra o consumo elétrico em um intervalo de dias ou apenas um dia dividido em horas.



Figura 10 – Pesquisa por um intervalo de DIAS o consumo [Google Charts, 2014].





**Figura 11 – Pesquisa por um determinado DIA dividido em HORAS de consumo[Google Charts, 2014].**

## 5. Implementação

A implementação do trabalho envolveu várias fases, iniciando com a montagem do circuito, aquisição do conjunto de peças, estrutura do código fonte no Arduino e do banco de dados e por fim a programação do gráfico de consumo energético, na qual foi feita utilizando a linguagem PHP e a *API Google Charts*, na qual exibe os resultados de maneira limpa e interativa [Google Charts, 2014].

### 5.1 Códigos Sensor.ino

O código implementado para este projeto refere-se ao arquivo *Sensor.ino* escrito na linguagem de programação Arduino, no qual é baseado em C++, que permite uma maior facilidade em operações com entrada e saída de dados [Arduino, 2013].

O sensor de corrente e o Ethernet Shield com slot para cartão SD, possuem códigos prontos que fazem parte da linguagem de hardware dos componentes, sendo apenas necessários somente incluir suas bibliotecas e definir suas variáveis [Openenergymonitor, 2014]. Exemplo:

```
#include <SD.h>           // Biblioteca do cartão SD.
#include <EmonLib.h>      // Biblioteca para comunicação Sensor.

EnergyMonitor emon1;    // Instância do monitor de energia da
                        // 0Biblioteca Emonlib.h.
const int CT_PIN = 1;  // Pino onde está conectado o sinal do sensor
                        // de corrente.
int segundo=0;         // Variavel utilizada no relógio.
int minuto=0;         // Variavel utilizada no relógio.
int hora=0;           // Variavel utilizada no relógio.
int dia=1;            // Variavel utilizada no relógio.
```

### 5.1.1 Sensor.ino função Setup()

Nesta etapa é feita a inicialização de componentes, na qual são instanciados portas analógicas e digitais. A verificação do slot SD no Ethernet Shield é feita neste momento. Caso haja algum erro ou ausência do cartão de memória, então o sistema é abortado e a operação loop() não é inicializada [Openenergymonitor, 2014].

```
void setup() {
  delay(1000);
  Serial.begin(9600);
  emon1.current(CT_PIN, 1.80);
  //emon1.voltage(2, 234.26, 1.7);

  Serial.print("Initializing SD card...");
  pinMode(10, OUTPUT);

  if (!SD.begin(4)) {
    Serial.println("initialization failed!");
    return;
  }
}
```

### 5.1.2 Sensor.ino função Loop()

No loop(), primeiramente é iniciado as funções da biblioteca Emonlib.h. O sistema a cada segundo vai escrever no arquivo leitura.sql os dados coletados com o sensor, que irá registrar o momento em hora, minuto e segundo de cada leitura, conforme o trecho de código abaixo [Openenergymonitor, 2014].

```
...
double Irms = emon1.calcIrms(1480);
double Potencia = Irms * 240.0;
...
myFile = SD.open("leitura.sql", FILE_WRITE);
if (myFile) {
  Serial.print("Escrevendo no arquivo leitura.sql... ");
  myFile.print("insert INTO `sensor`.`leitura` (`irms`, `watts`, `dia`,
`hora`, `minuto`, `segundo`) VALUES ('");
```

### 5.3 Método emon1.calcIrms()

Nesta etapa, é feita a coleta de valores do sensor de energia [Openenergymonitor, 2014].

```
class EnergyMonitor
{
public:
  void voltage(int _inPinV, double _VCAL, double _PHASECAL);
  void current(int _inPinI, double _ICAL);
  void voltageTX(double _VCAL, double _PHASECAL);
  void currentTX(int _channel, double _ICAL);
  void calcVI(int crossings, int timeout);
  double calcIrms(int NUMBER_OF_SAMPLES);
  void serialprint();
  long readVcc();
  double realPower,
  apparentPower,
  powerFactor,
  Vrms,
  Irms;
}
```

## 5.4 Gráfico Index.php

Os conteúdos do arquivo index.php é dividido em 2 etapas, a primeira tem como função, ler as informações que estão no banco de dados e armazenando-as em variáveis. A segunda é utilizar estas variáveis para repassa-las a uma API, conhecida como *API Charts*, da empresa *Google Inc.* Este por sua vez, é responsável por interpretar os dados e exibi-los em forma de gráficos. Abaixo segue um trecho da *API Google Charts* sem as modificações realizadas para este trabalho [Google Charts, 2014].

```
google.load("visualization", "1", {packages:["corechart"]});
google.setOnLoadCallback(drawChart);
function drawChart() {
    var data = google.visualization.arrayToDataTable([
        ['Year', 'Sales', 'Expenses'],
        ['2013', 1000, 400],
        ['2016', 1030, 540]]);
    var options = {
        title: 'Company Performance',
        hAxis: {title: 'Year', titleTextStyle: {color: '#333'}},
        vAxis: {minValue: 0}
    };

    var chart = new
    google.visualization.AreaChart(document.getElementById('chart_div'));
    chart.draw(data, options);
}
```

O mesmo arquivo fornece 2 maneiras de pesquisa, uma em forma de intervalos de dias, e outra em forma de horas. Para isto foi utilizado ferramentas de linguagem HTML mesclando com ferramentas de linguagem PHP, conforme mostra o trecho abaixo.

```
<td width="40"><select name="minutoi" id="minutoi">
<?php
    do { ?>

<option value="<?php echo $row_auth2['dia']; ?>">
    <?php echo $row_auth2['dia']; ?>
</option>
<?php }
while ($row_auth2 = mysql_fetch_assoc($auth2));
mysql_free_result($auth2);
?></select></td>
```

## 6. Metodologia

O modelo utilizado para este trabalho é conhecido como clássico ou cascata pois possui uma abordagem “*top-down*”, no qual foi indicado por *Royce* no ano de 1970 sendo conhecido como um paradigma padrão no desenvolvimento de softwares e na criação de novos modelos (SOMMERVILLE, 2004).

O modelo em cascata, que também é conhecido por alguns autores como clássico, ciclo de vida clássico, ciclo de vida de software, entre outros, constitui-se em um modelo mais antigo e simples de desenvolvimento e criação de software. Como particularidade ele pode assumir uma ordenação linear e seqüencial, onde uma etapa esta conectada a outra, a saída da anterior está ligada na entrada da próxima assim seguindo o padrão, tal como mostrado na Figura 15 (PRESSMAN, 2006)

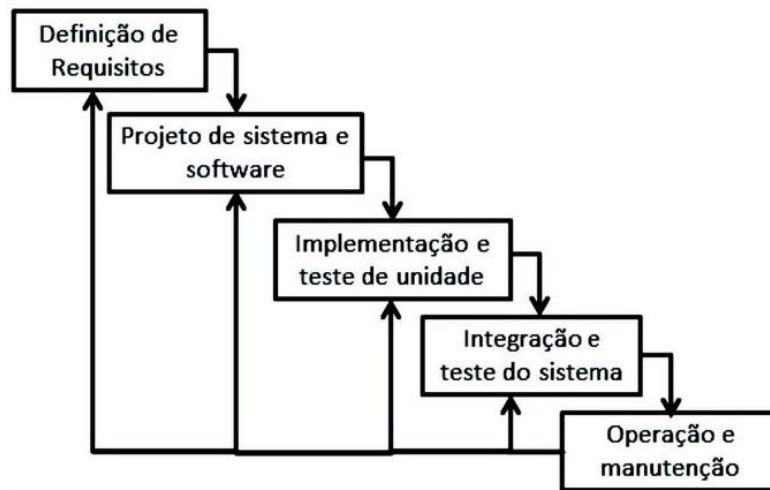


Figura 12 – Modelo Cascata (SOMMERVILLE, 2004)

### 6.1 Requisitos Funcionais

Os requisitos funcionais (RF) tratam de descrever qual deve ser o comportamento do sistema a partir da definição de suas funcionalidades. Normalmente estão representados no diagrama de casos de uso (SOMMERVILLE, 2008).

Requisito Funcional	Descrição	Caso de Uso
Leitura de tensão	O sensor deve realizar a leitura de energia elétrica.	Leitura de tensão
Armazenar dados	O Arduino deve armazenar os dados lidos pelo sensor em um cartão de memória.	Armazenar dados
Transferência dos dados	Os dados coletados no cartão de memória serão transferidos e armazenados em um banco de dados.	Transferir dados
Exibir Resultado	Através dos dados contidos no banco de dados, será gerado planilhas e gráficos, com informações referentes ao tempo de leitura.	Exibir Resultados

Tabela 4 – Requisitos Funcionais

### 6.2 Requisitos não funcionais

Os requisitos não funcionais (RNF) são relacionados com as funções específicas do sistema. Podem estar relacionados com propriedades emergentes como, por exemplo, confiabilidade, tempo de resposta, especificando ou restringindo estas propriedades (SOMMERVILLE, 2008).

Requisito Não Funcional	Descrição
Utilização de Sistema Operacional	O sistema operacional utilizado para o desenvolvimento do software será o Microsoft <i>Windows 7</i> tendo possibilidade de portabilidade para linux.
Utilização de um cartão de memória	Será utilizado um cartão de memória, no qual será responsável por armazenar os dados gerados pelo Arduino.

Utilização de um banco de dados	O banco de dados utilizado é o <i>MySQL</i> , pois tratando-se de um banco de dados relacional, consegue satisfazer todas as necessidades.
---------------------------------	--

Tabela 5 – Requisitos Não Funcionais

### 6.3 Diagrama de Caso e Uso

Abaixo segue o diagrama de caso de uso, Figura 16, que descreve as comunicações do sensor de tensão com o microcontrolador.

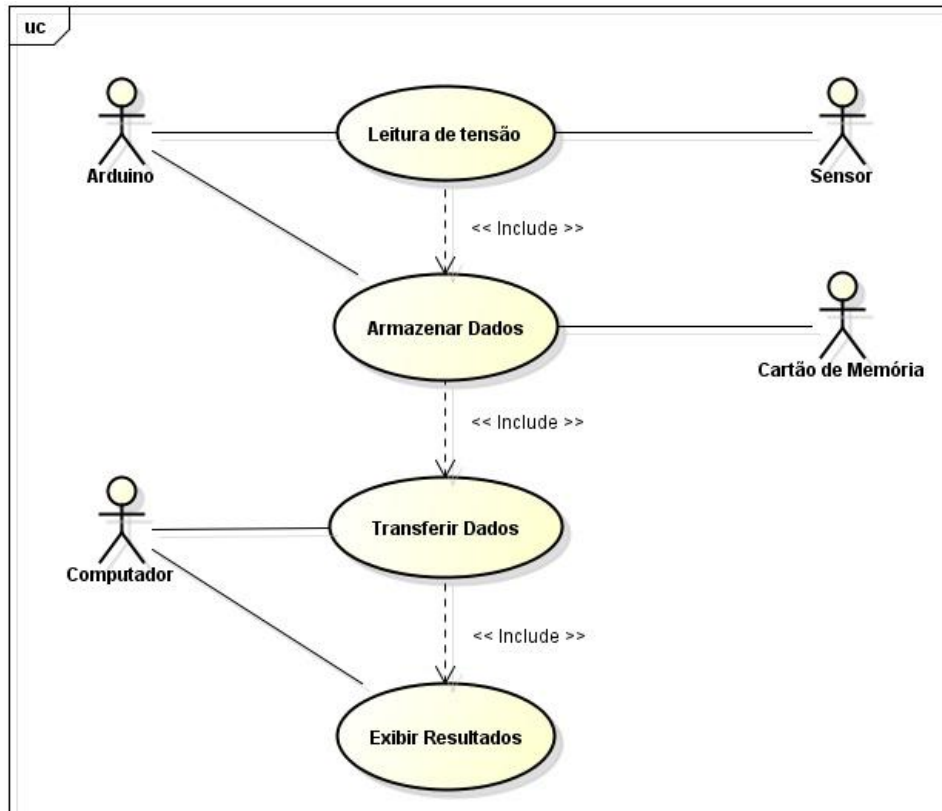


Figura 16 – Diagrama de Caso de Uso

### 6.3 Diagrama de Atividades

O diagrama de atividades, Figura 17, pode ser observado como o dispositivo se comporta após ler uma tensão vinda do sensor. Dá início ao processo com a adaptação do sensor na rede elétrica, é verificado se o cartão de memória está inserido no Arduino, caso sim, é feita a leitura da energia elétrica e armazenado os dados no cartão de memória, quando parado a leitura, o sensor é desconectado da corrente elétrica, o cartão de memória é retirado do Arduino, e transferido para computador onde os dados serão inseridos no banco de dados, desta maneira é possível gerar informações em formato de planilha sobre a leitura feita da corrente elétrica, caso a leitura continue o sistema volta a inserir o cartão de memória no Arduino iniciando novamente o ciclo caso não haja mais leitura o sistema é finalizado.

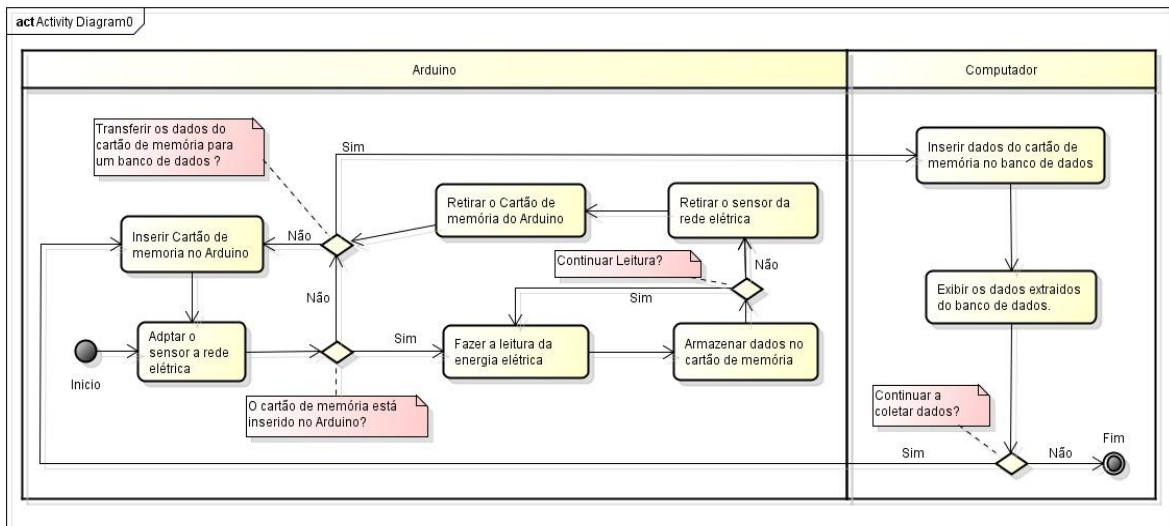


Figura 17 – Diagrama de Atividades

### 6.3 Diagrama de Classes

Através do diagrama de classes pode ser apresentado as relações e estruturas das classes. Neste modelo, são analisados os objetos das classes, suas penalidades, relacionamentos entre outros (SOMMERVILLE, 2008).

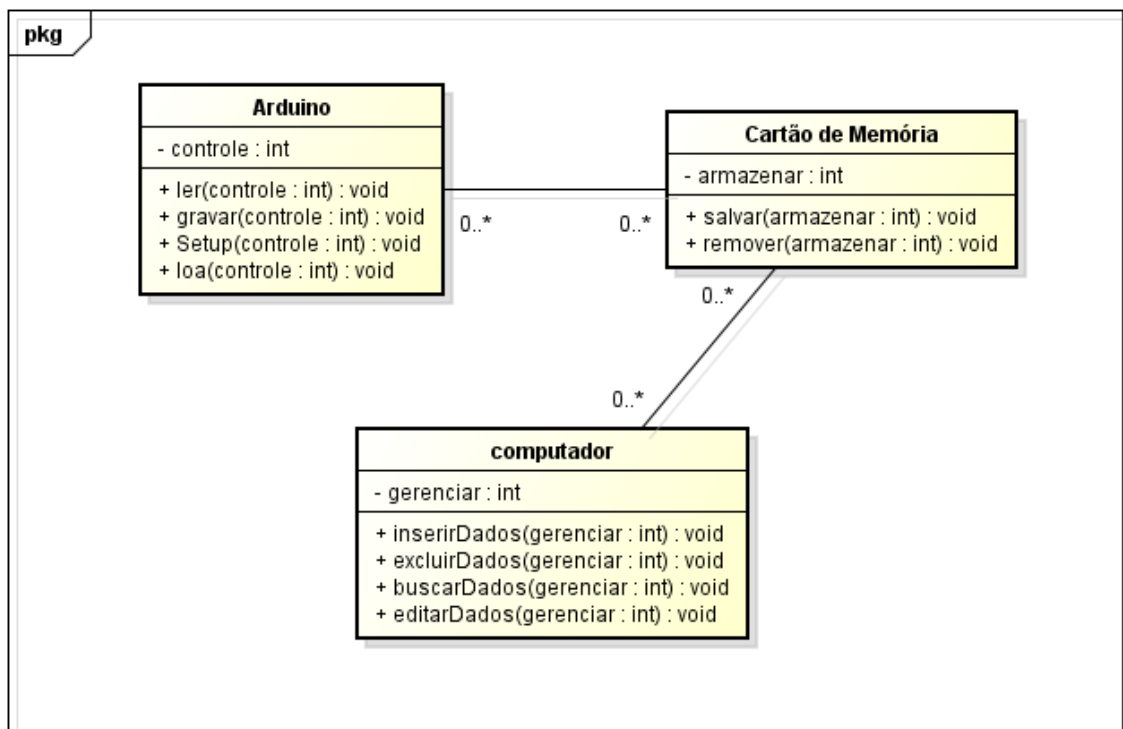


Figura 18 – Diagrama de Atividades

## 7. Testes

O sensor foi testado primeiramente em lâmpadas incandescentes, os valores obtidos foram muito próximos ao que indicava em suas respectivas potencias, chegando até a valores idênticos, tendo em vista que o sensor foi calibrado para uma frequência de

voltagem 220v. O sensor teve uma eficiência mínima quando testados em correntes opostas, não obtendo assim nenhuma leitura significativa.

O sensor não soube medir cabos onde juntos estejam o par positivo e negativo no mesmo fio. Desta forma os cabos negativos e positivos tendem a se anular, resultando numa leitura zerada, para isso foi necessário corta-los e escolher ou cabo positivo ou no cabo negativo para adaptar ao sensor.

O sensor obteve sucesso ao ser instalado na fase final de uma empresa ou indústria, recebendo o acumulado total de energia consumida por diversos aparelhos nela ligados, para obter a informação de consumo dos aparelhos individualmente, foi necessário desliga-los um a um e subtrair a carga total menos o atual, obtendo assim a diferença entre elas, resultando na carga individual do mesmo.

O sensor obteve sucesso para ler qualquer tipo de potência sendo limitado apenas pelo diâmetro do cabo, onde o mesmo não podendo passar de 13 mm.

## 8 Conclusão

Este trabalho utilizou conceitos e técnicas estudadas, a fim de alcançar o objetivo, que foi a criação de um sensor capaz de capturar e gravar os dados de campos eletromagnéticos que acompanham a corrente elétrica e variam de acordo com sua potência. Contando com o apoio de diversas tecnologias disponíveis com o conjunto destas teorias tornou-se possível a criação e desenvolvimento do projeto.

No início ocorreram dificuldades de configuração do hardware, havendo um maior estudo sobre o funcionamento e interação, bem como interpretação dos componentes. Estas por sua vez foram sanadas, pois existe um vasto material em websites ou livros, que fornecem embasamento para montagem do sensor.

Foi fundamental que para o entendimento, a necessidade do estudo de algumas grandezas físicas, assim com tensão, corrente elétrica, potencia, além da assimilação de teorias.

A coleta dos dados ficou responsável pelo equipamento adquirido na própria internet, assim como seus componentes. A tecnologia usada para o banco de dados e a geração do gráfico de consumo, é livre e gratuita, os códigos criados para o projeto são um misto destas tecnologias amplamente difundidas.

Por fim este trabalho desenvolveu um sensor capaz de auxiliar o monitoramento de energia elétrica consumida, para que seus comportamentos possam ser coletados, analisados e interpretados. Deste modo o trabalho visou acrescentar um novo componente que possa agregar no monitoramento da energia elétrica, em um futuro que a energia consumida será cobrada por horários do dia, sendo possível, utilizar esta energia de forma mais econômica e direcional.

## 9. Referências

ARDUINO. **Arduino**. Disponível em <<http://www.arduino.cc>>. Acesso em maio de 2013.

ALEXANDER, K. Charles; SADIKU, O. N. Matthew. **Fundamentos de Circuitos Elétricos**. Porto Alegre, RS: Bookman 2003.

ALVARENGA, Beatriz; MÁXIMO, Antônio. **FÍSICA VOLUME 3** : ensino médio. São Paulo, SP: Editora Scipione, 2006.

DATE, C. J. **Introdução a Sistemas de Banco de Dados**. 8ª edição. RJ: Editora Elsevier, 2003. Disponível em <<http://books.google.com.br/books?hl=pt-BR&lr=&id=xBeO9LSIK7UC&oi=fnd&pg=PP23&dq=banco+de+dados&ots=x9SAe-Ag3O&sig=LpOMVtFKZ6rITIMQl ynFxmD7z9U>>. Acesso em julho de 2013.

GOOGLE CHARTS. Google Gráficos. Disponível em <<https://google-developers.appspot.com/chart/interactive/docs/gallery/areachart>>. Acesso em maio de 2014.

MOSSER, Vicent, **United Stats Patent** 5,442,221, 15 Agosto 1995, Hall Efect Sensor.

MLEMOS. Medidor de Consumo de Energia Elétrica Conectado à Nuvem com Arduino. Disponível em <<https://github.com/mlemos/energy-monitor-cpbr7>>. Acesso em maio de 2014.

PRESSMAN, Roger S. **Engenharia de Software**. 6ª edição., RJ: McGraw-Hill, 2006.

SOMMERVILLE, Ian. **Engenharia de Software**. 6ª edição. São Paulo, SP: Addison Wesley, 2003.