UNIVERSIDADE FEDERAL FLUMINENSE

FELIPE SOUZA GUEDES

Internet das Coisas (*IoT – Internet of Things*) - Implementação de um medidor de energia elétrica com conexão a Internet.

Niterói

2018

FELIPE SOUZA GUEDES

Internet das Coisas (*IoT – Internet of Things*) - Implementação de um medidor de energia elétrica com conexão a Internet.

Trabalho de Conclusão de Curso submetido ao Curso de Tecnologia em Sistemas de Computação da Universidade Federal Fluminense como requisito parcial para obtenção do título de Tecnólogo em Sistemas de Computação.

Orientador(a):
Julliany Sales Brandão

NITERÓI
2018

Ficha catalográfica automática - SDC/BEE

G924i Guedes, Felipe Souza
Internet das Coisas (IoT - Internet of Things) - Implementação de um medidor de energia elétrica com conexão a Internet. / Felipe Souza Guedes ; Julliany Sales Brandão, orientadora. Niterói, 2018.
69 f. : il.

Trabalho de Conclusão de Curso (Graduação em Tecnologia de Sistemas de Computação)-Universidade Federal Fluminense, Escola de Engenharia, Niterói, 2018.

1. Informática. 2. Internet das Coisas. 3. Banco de Dados. 4. Sistemas de Computação. 5. Produção intelectual. I. Título II. Brandão,Julliany Sales , orientadora. III. Universidade Federal Fluminense. Escola de Engenharia. Departamento de Ciência da Computação.

CDD -

Bibliotecária responsável: Fabiana Menezes Santos da Silva - CRB7/5274

FELIPE SOUZA GUEDES

Internet das Coisas (*IoT – Internet of Things*) - Implementação de um medidor de energia elétrica com conexão a Internet.

Trabalho de Conclusão de Curso submetido ao Curso de Tecnologia em Sistemas de Computação da Universidade Federal Fluminense como requisito parcial para obtenção do título de Tecnólogo em Sistemas de Computação.

Niterói, ___ de _____ de 2018.

Banca Examinadora:

_____

Profª. Julliany Sales Brandão, DSc.. – Orientadora

CEFET/RJ - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

_____

Prof. Flávio Luiz Seixas, DSc. – Avaliador

UFF – Universidade Federal Fluminense

Dedico este trabalho a minha esposa, aos meus estimados pais e a todos que compartilham o conhecimento.

AGRADECIMENTOS

A Deus, que sempre iluminou a minha caminhada.

A minha orientadora Julliany Sales Brandão pelo estímulo e atenção que me concedeu durante o curso.

Aos Colegas de curso pelo incentivo e troca de experiências.

A todos os meus familiares e amigos pelo apoio e colaboração.

“Nenhum homem realmente produtivo pensa como se estivesse escrevendo uma dissertação. ”

Albert Einstein

RESUMO

A Internet das Coisas (*IoT* do inglês *Internet of Things*) é uma revolução tecnológica com o objetivo de conectar os itens do cotidiano a Internet, transformando o mundo físico e digital em um único. Este tema tem sido bastante estudado na literatura e tem apresentado resultados interessantes, como sua implementação nos mais variados itens. Neste trabalho será utilizado a *IoT* para medir o consumo de energia elétrica em uma residência com o objetivo do controle do consumo com a maior facilidade e disponibilidade dos dados em qualquer lugar a partir de um site na Internet. As tecnologias envolvidas para a construção do trabalho foram a *plataforma Arduino*; linguagens de programação C, PHP e HTML; e banco de dados SQL. Os resultados mostraram que a *IoT* facilitou o acompanhamento do consumo de energia elétrica de uma residência e que as tecnologias disponíveis para sua implementação são de uso bastante simplificado.

Palavras-chaves: Internet das Coisas, Arduino, Consumo de energia.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADSL – Asymmetric Digital Subscriber Line
API – Application Programming Interface
DNS – Domain Name System
ECG – Eletrocardiograma
EEPROM – Electrically-Erasable Programmable Read-Only Memory
HTML – HyperText Markup Language
ICSP – In-Circuit Serial Programming
IDE – Integrated Development Environment
IEEE – Institute of Electrical and Electronics Engineers
IoT – Internet of Things
IP – Internet Protocol
PHP – PHP: Hypertext Preprocessor
PWM – Pulse Width Modulation
RAM – Random Access Memory
SGBD – Sistemas de Gerenciamento de Banco de Dados
SoC – System on Chip
SQL – Structure Query Language
SRAM – Static Random Access Memory
TC – Transformador de Corrente
TCP/IP – Transmission Control Protocol / Internet Protocol
UART – Universal Asynchronous Receiver/Transmitter
USB – Universal Serial Bus
Wh – Watt hora

SUMÁRIO

1 INTRODUÇÃO

A Internet está começando a comandar diretamente o hardware em vez de comandar apenas os navegadores e os servidores web. Essa é era da chamada Internet das Coisas (IoT do inglês: *Internet of Things*) [1]. “A IoT se refere a uma revolução tecnológica que tem como objetivo conectar os itens usados no dia a dia à rede mundial de computadores. Cada vez mais surgem eletrodomésticos, meios de transporte e até mesmo tênis, roupas e maçanetas conectadas à Internet e a outros dispositivos, como computadores e *smartphones*. A ideia é que, cada vez mais, o mundo físico e o digital se tornem um só, através de dispositivos que se comuniquem com os outros, os data-centers e suas nuvens” [2]. Cada vez mais encontra-se os conceitos da Internet das Coisas no nosso dia-a-dia, como por exemplo, carros com monitoramento e acionamento a distância, refrigeradores com acesso a Internet, monitoramento de sinais cardíacos durante exercícios, entre muitos outros. A Internet das Coisas possui ainda muitas oportunidades de crescimento, o que motiva a realização de um trabalho nessa área.

O trabalho consiste em pesquisar, implementar e analisar a construção de um medidor de consumo de energia elétrica, enviando dados para um banco de dados na Internet, com intuito de acompanhar o consumo de uma residência. Esse acompanhamento, se feito de forma manual, necessitaria que o interessado fizesse medições em intervalos de tempo regulares, deslocando-se ao local da instalação do medidor da concessionária de energia elétrica e anotando os valores para o histórico. Com o medidor enviando dados para a “nuvem”, além de não precisar ir até o local da medição, poderá ter mais medições por intervalo de tempo e ainda acesso as informações em qualquer lugar utilizando um computador ou *smartphone* conectado a Internet. A conexão do medidor por redes sem fios facilita consideravelmente a instalação, não precisando de passagem de cabos e conexões elétricas.

Para a implementação do medidor será utilizado a *plataforma Arduino* para coleta dos dados físicos e envio para a Internet através de uma rede sem fios. Nessa parte, conceitos de programação em *linguagem C*, organização de

computadores e redes serão muitos utilizados. Após os dados serem enviados para a Internet, eles deverão ser tratados e armazenados em um banco dados, para isso conhecimentos em linguagem de *programação PHP, banco de dados SQL e linguagem HTML* serão aplicados.

O trabalho será organizado da seguinte forma: no Capítulo 2 será tratado o projeto de forma macroscópica, analisando e definindo os requisitos a serem atendidos, funcionamento geral e equipamentos necessários. O Capítulo 3 é dedicado a *plataforma Arduino* e como foi utilizado para a implementação. No Capítulo 4 é mostrada a configuração e programação do servidor Web para recebimento e armazenamento dos dados. Os resultados e as análises serão mostrados no Capítulo 5 e finalmente no Capítulo 6 serão apresentadas as conclusões e indicações para trabalhos futuros.

2 O MEDIDOR DE ENERGIA ELÉTRICA

2.1 APRESENTAÇÃO

O projeto consiste em implementar um medidor de consumo de energia elétrica, enviando dados para um banco de dados na Internet, com intuito de acompanhar o consumo de uma residência. Para fazer esse acompanhamento, pode-se verificar o consumo na conta de energia elétrica. Dessa forma tem-se o acompanhamento com uma frequência mensal e com nenhum detalhamento de horário diário de maior consumo ou numa determinada época do mês. Outra forma seria ir com certa frequência até o medidor da concessionária de energia elétrica e anotar o valor de consumo atual. Nesse caso, o nível de detalhamento está diretamente ligado a quantidade de medições num espaço de tempo e as anotações para calcular o consumo no período. Essa última forma faz com que o interessado dispense de muitos recursos (tempo, deslocamento, anotações) para ser ter uma informação de boa qualidade.

O objetivo é ter um medidor em funcionamento em paralelo com o da concessionária. Esse medidor coletará os dados e enviará para um banco de dados com uma frequência adequada, e depois os dados serão disponibilizados para o acesso em qualquer lugar através de uma página na Internet.

O intuito é utilizar uma conexão com a Internet através de uma rede sem fio doméstica, hoje muito comum nas residências no Brasil. A verificação dos dados será feita através de acesso a uma página da Internet desenvolvida para esse fim.

Essa ideia de medidores conectados a Internet transmitindo dados de consumo pode além de trazer comodidade e um maior controle do consumo para o próprio consumidor, pode trazer benefícios para até mesmo à concessionária de energia elétrica. No caso do consumidor, ele pode verificar se seu gasto mensal está ultrapassando o planejado em seu orçamento financeiro, assim adequando o seu consumo. O consumidor ainda pode ser avisado no caso de um aumento anormal de consumo causado por alguma falha de algum equipamento ou da instalação elétrica (por exemplo, baixa isolação) podendo atuar na solução do problema o quanto

antes. No caso da concessionária de energia elétrica, ao invés de enviar um funcionário a cada cliente em certo dia do mês, poderia obter seu consumo remotamente. A medição remota também não teria o problema de leitura ou registro errado da medição, ou falta de acesso ao medidor devido a ausência do consumidor, que causam trabalho adicional à concessionária.

Outro ponto importante com o armazenamento dos dados de consumo dos usuários é ter dados mais precisos para analisar o perfil de consumo de energia elétrica com intuito de auxiliar definições nas políticas de investimentos.

Um trabalho similar com base similar foi apresentado em [3], onde um monitor de sinais de ECG (Eletrocardiograma) envia sinais para um banco de dados remoto para posterior análise e acompanhamento. Nesse trabalho são utilizadas ferramentas já prontas para o armazenamento dos dados, como o *ThingSpeak* [4]. A implementação proposta além de diferir no sinal captado, consumo de energia ao invés de sinais de batimentos cardíacos, também terá o desenvolvimento do sistema próprio de armazenamento dos dados. Para o hardware será utilizado como base a proposta do site *www.openenergymonitor.org* [5], que é um projeto para implementação de um medidor de energia elétrica.

2.2 DIAGRAMA DE BLOCOS

Para um amplo entendimento, o projeto será dividido em partes representado pelo diagrama de blocos apresentado na Figura 1. A medição de energia elétrica será realizada por um sensor de corrente (1). O sinal da medição (a) é então enviado para um microcontrolador (2) que calculará o consumo de energia. Após esse cálculo, os dados são enviados via comunicação serial (b) para serem transmitidos para a Internet no módulo de comunicação sem fios (3). O módulo de comunicação sem fios se conectará a Internet (5) por meio de um roteador (4). Os dados serão armazenados num servidor (6), para posteriormente serem acessados em computadores, *smartphones* ou outros dispositivos com acesso a Internet (7).

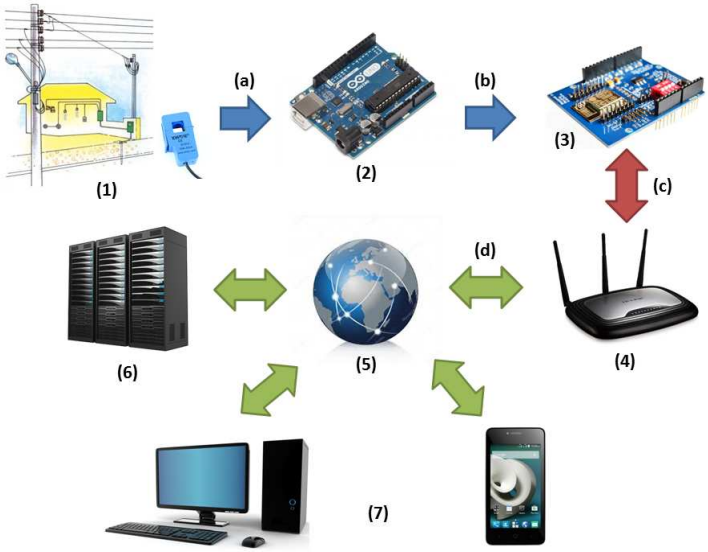


Figura 1: Diagrama de blocos do projeto.

2.2.1 Medição de consumo de energia elétrica

Normalmente na entrada de energia elétrica em uma residência, o primeiro componente é o medidor de consumo de eletricidade da concessionária. O consumo de energia é medido em Watts-horas (Wh), que é uma relação entre a tensão elétrica, corrente elétrica e tempo. Essa relação é dada por:

$$EnergiaElétrica = Potência \text{ x } Tempo \quad (1)$$

sendo:

$$Potência = Tensão \text{ x } Corrente \quad (2)$$

Substituindo a equação (2) em (1), tem-se:

$$Energia\ Elétrica = Tensão \text{ x } Corrente \text{ x } Tempo \quad (3)$$

Analisando a equação (3) verifica-se que para a medição do consumo de energia elétrica é necessário o monitoramento de três variáveis: tensão, corrente e tempo.

Para o projeto será considerado o valor de tensão fixo em 127V que é a tensão nominal da concessionária. A corrente será monitorada por um sensor de

corrente elétrica, conforme a Figura 2. Esse sensor funciona basicamente da seguinte forma: quando uma corrente elétrica passa por um condutor é gerado um campo magnético que induz uma corrente elétrica em uma bobina gerando uma tensão proporcional a corrente medida. Com relação ao tempo, este será calculado com base na frequência de operação do microcontrolador utilizado no projeto.

Figura 2: Sensor de corrente elétrica.[1]

2.2.2 Processamento dos sinais

Para o processamento dos sinais, é utilizado uma plataforma de prototipagem conhecida como *Arduino*. Ela é responsável pela leitura do sinal proveniente do sensor de corrente através das suas entradas analógicas, cálculo da potência com base na corrente medida e integrá-la com relação ao tempo para obter a consumo de energia elétrica. Outra função ainda é enviar o dado de consumo para a Internet através do módulo de rede sem fio.

O módulo de rede sem fio comunica-se com o *Arduino* através de uma conexão serial. Com a rede sem fio para acesso a Internet ele utiliza o protocolo IEEE 802.11g para conexão com o roteador. A partir do roteador até a Internet, o fluxo não é uma questão relevante para o projeto, pois isso pode ser feito de várias formas diferentes como através de modem ADSL, via cabo de TV por assinatura, fibra óptica, entre outros, ficando totalmente transparente ao projeto.

[1] Fonte: http://www.yhdc.us

2.2.3 Armazenamento e disponibilização dos dados

Através da Internet, os dados enviados serão recebidos e armazenados em um servidor. Esse servidor será um servidor operando com linguagem PHP e um banco de dados MySQL.

Os usuários poderão acessar os dados do servidor também através da Internet, utilizando um navegador de Internet a partir de um microcomputador ou *smartphone*, acessando uma página da Internet.

3 PLATAFORMA ARDUINO

3.1 O QUE É O ARDUINO?

De acordo com próprio site o *Arduino* [6] é uma plataforma de prototipagem eletrônica de código aberto baseada em hardware e software de fácil utilização. Ela surgiu na Itália no *Ivrea Interaction Design Institute*.

Inicialmente o *Arduino* foi desenvolvido para estudantes sem experiência em eletrônica e programação, mas com o passar do tempo a comunidade de uso se tornou mais ampla e novas necessidades e desafios apareceram, aumentando a gama de itens disponíveis. Atualmente a plataforma recebe contribuições de usuários de todo o mundo.

Como em um computador, a plataforma Arduino pode ser dividida em dois itens: hardware e software.

O hardware da plataforma Arduino é baseado em microcontroladores *Atmel AVR*. Em geral são microcontroladores de 8 bits, mas atualmente já existem versões com microcontroladores de 32 bits. Na Figura 3, são apresentados exemplos de duas placas *Arduino*, a versão UNO e a MEGA.

Para a programação do *Arduino* é utilizado uma linguagem de programação baseada em *C++*, mas também pode ser utilizado a linguagem *AVR C* (linguagem com as instruções do microcontrolador). Toda programação é feita num ambiente de programação conhecido como *Arduino IDE*, disponível para vários sistemas operacionais e também com uma opção online (*Arduino Web Editor*), com uma vasta biblioteca de programas mantida por grupos de usuários, que facilita muito a sua utilização.

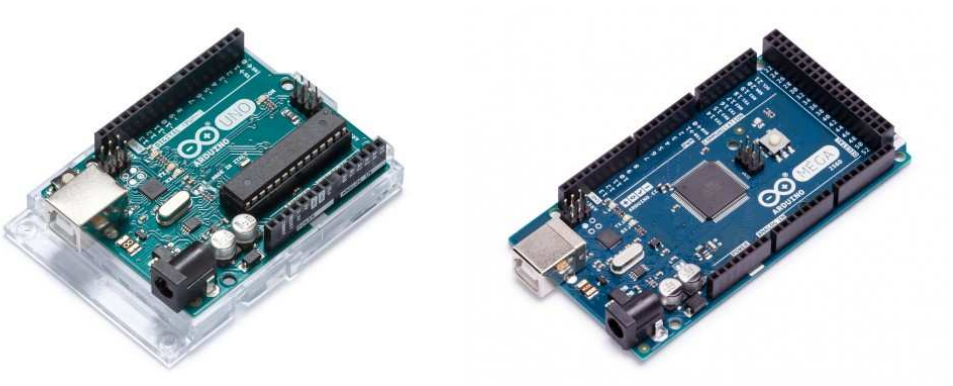


Figura 3: Arduino UNO e Arduino MEGA.[2]

A plataforma Arduino possui muitas vantagens como:

- Baixo custo – o hardware é relativamente barato em comparação com outras plataformas.
- Multiplataforma – o software do Arduino pode ser executado em diversos sistemas operacionais como *Windows, Macintosh OSX* e *Linux*.
- Ambiente de programação simples e claro - o software *Arduino* é fácil de usar para iniciantes, e também flexível o suficiente para usuários avançados.
- Software de código aberto e extensível – o software é publicado como ferramenta de código aberto, sendo possível alterações por usuários experientes. Ele é extensível pois pode ser utilizado com bibliotecas da linguagem de programação C++, ou mesmo com linguagem de programação AVR C (linguagem da plataforma do microcontrolador em que o Arduino se baseia).
- Hardware de código aberto e extensível – todo o circuito eletrônico é publicado na íntegra e pode ser modificado para a adequação da necessidade.

A plataforma *Arduino* ainda conta com placas de expansões, conhecidas como *Shields*, que agregam outras funções como redes de comunicação, displays, saídas para controle de motores, expansão de entradas e saídas, entre outras. Na Figura 4 é mostrado um *Arduino UNO* com um *Shield* de rede *Ethernet*.

[2] Fonte: www.arduino.cc

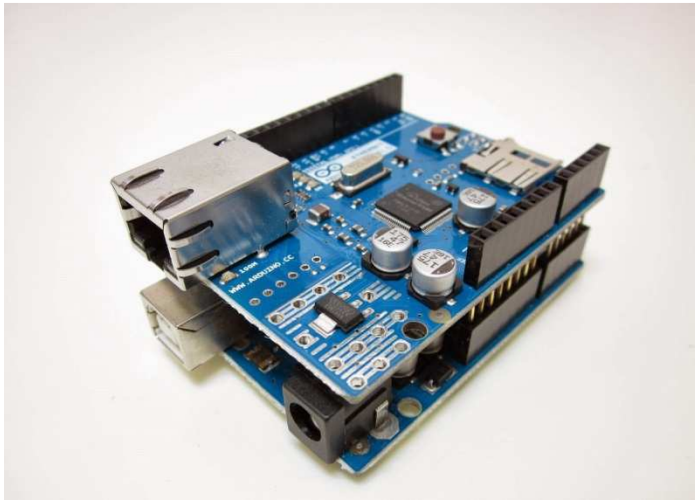


Figura 4: Arduino UNO com *Shield Ethernet*.[3]

3.2 O ARDUINO UNO

O Arduino UNO é uma placa baseada no microcontrolador ATmega328P [7]. Dispõe de 14 pinos digitais de entrada / saída, 6 dos quais podem ser utilizados como saídas PWM (*Pulse Width Modulation*), 6 entradas analógicas, um cristal de quartzo 16 MHz, de uma entrada USB para comunicação e programação, um conector para alimentação, um conector ICSP (*In Circuit Serial Programming*) e um botão de Reset. Na Figura 5 pode-se ver a localização desses itens. O diagrama esquemático do circuito eletrônico é mostrado no ANEXO A. A Tabela 1 reúne as especificações do *Arduino UNO*.

[3] Fonte: http://www.facacomarduino.info

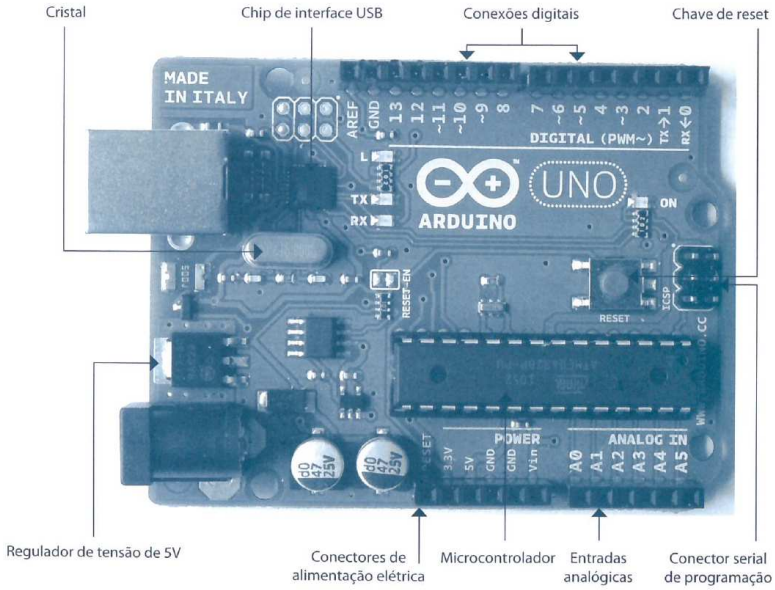


Figura 5: Componentes do Arduino UNO.[4]

Tabela 1: Especificações Arduino UNO

Microcontrolador	ATmega328P
Tensão de Operação	5V
Tensão de Entrada (recomendada)	7-12V
Tensão de Entrada (limite)	6-20V
Pinos de Entrada/Saída Digitais	14 (das quais 6 podem ser sáidas PWM)
Pinos de Entradas Analógicas	6
Corrente por pino de Entrada/Saída Digital	20 mA
Corrente para pino de 3,3V	50 mA
Memória Flash	32 KB (ATmega328P) dos quais 0.5 KB são usados pelo *bootloader*
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Frequência de Clock	16 MHz

Para a implementação do projeto, tem-se como requisitos:

- 1 entrada analógica para a medição de corrente elétrica.

[4] Fonte: Programação com Arduino: começando com Sketches.

- 2 pinos de Entrada/Saída para a comunicação com a placa de rede sem fio.
- 1 porta USB para *“Debug”* e acompanhamento do sistema.

Portanto a plataforma Arduino UNO atenderá os requisitos, sendo adequada para implementação do projeto.

3.3 CIRCUITO DO SENSOR DE CORRENTE ELÉTRICA

Para a medição da corrente elétrica consumida, será utilizado um sensor de corrente. Esses sensores são também conhecidos como transformadores de corrente (TC). Como qualquer outro transformador, um transformador de corrente possui um enrolamento primário, um núcleo magnético, e um enrolamento secundário. O enrolamento primário nesse caso será o condutor onde se deseja medir a corrente, o núcleo magnético é o responsável por transmitir o fluxo magnético para o enrolamento secundário, e o enrolamento secundário é feito de muitas voltas de fio fino no núcleo.

A corrente alternada que flui no enrolamento primário produz um campo magnético no núcleo, que induz uma corrente no circuito do enrolamento secundário. A corrente no enrolamento secundário é proporcional à corrente que flui no enrolamento primário:

O número de espiras secundárias no TC é dependente de sua construção. Normalmente, essa proporção está escrita em termos de correntes ou em termos de corrente e tensão quando possuir um resistor de carga incluso. No caso do projeto será utilizado um TC com relação de 0-50A/0-1V, ou seja, para uma corrente de 50A tem-se uma saída de 1V.

Para se conectar o sensor de corrente elétrica a plataforma *Arduino*, o sinal do sensor de corrente deve ser condicionado de modo que satisfaça os requisitos das entradas analógicas. No caso do *Arduino UNO* que será utilizado precisa-se de uma tensão positiva entre 0V e a tensão de referência do conversor analógico digital que é 5V. A folha de dados do sensor está no Anexo C.

Baseado no site *www.openenergymonitor.org* [5], um circuito sugerido para essa finalidade é apresentado na Figura 6.

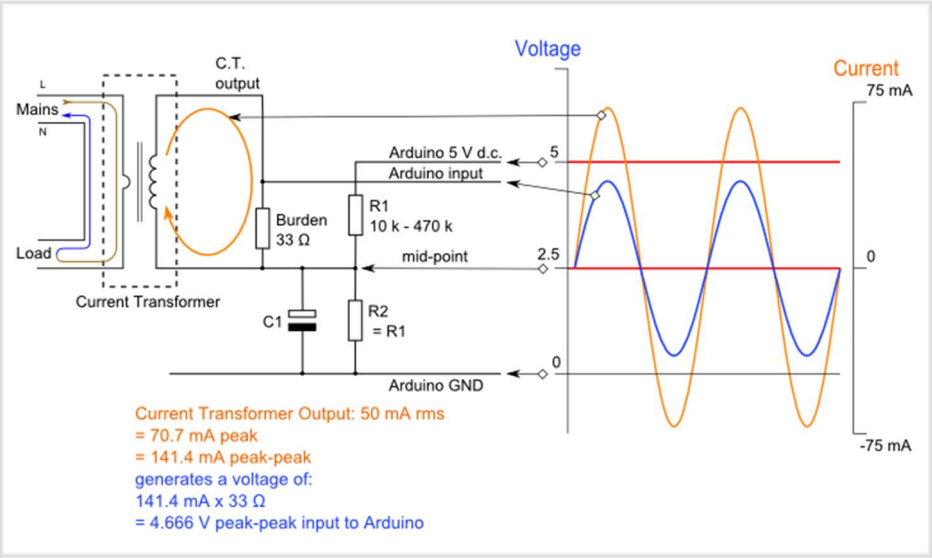


Figura 6: Circuito para acoplamento do sensor de corrente.[5]

Como já visto, uma característica da plataforma *Arduino* é a sua expansão através de placas. Para evitar a necessidade montagem de circuitos adicionais será utilizada uma placa de expansão (*Shield*) já com o circuito de acoplamento do sensor de corrente. Uma placa disponível para essa finalidade é a da fabricante *ElectroDragon* mostrada na Figura 7. O diagrama esquemático dessa placa é mostrado no Anexo B.

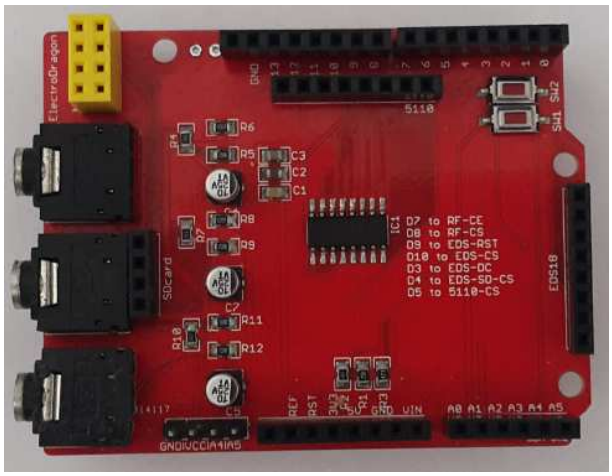


Figura 7: Placa para ligação dos sensores de corrente elétrica.[6]

[5] Fonte: *www.openenergymonitor.org*.

3.4 COMUNICAÇÃO *WIRELESS*

Uma rede sem fio (*wireless*) é uma estrutura de comunicação que permite a troca de dados e informações sem a necessidade de uso de uma conexão física (cabos elétricos ou fibras ópticas). Redes sem fio em geral usam de sinais de radiofrequência ou raios infravermelhos. A tecnologia das redes sem fios tem como sua principal vantagem a redução do custo de instalação, pois não há necessidade de instalação e ligações de cabos entre os dispositivos.

Para a interoperabilidade dos dispositivos sem fios, entidades como o IEEE (*Institute of Electrical and Electronics Engineers*) padronizam essas redes. Um dos padrões mais difundidos atualmente é o IEEE 802.11. Esse padrão possui algumas variações ocorridas com o avanço do tempo, em função da frequência de operação da rede, largura de banda, velocidade de transferência, alcance, entre outros. Em geral os roteadores de uso doméstico utilizam esse padrão.

No projeto, a interligação do circuito de medição a Internet para a disponibilização dos dados será feita através de uma placa de expansão para rede sem fio baseada no módulo ESP8266. A Figura 8 mostra a placa utilizada no projeto.

Figura 8: Placa de expansão módulo ESP8266. [7]

O módulo ESP8266 é um *SoC* (*System on Chip*) autônomo com pilha de protocolo TCP/IP integrada que pode dar acesso à rede sem fio [8]. Esse módulo recebe comandos via comunicação serial UART (*Universal Asynchronous*

[6] Fonte: Fotografia da placa utilizada na implementação.
[7] Fonte: Fotografia da placa utilizada na implementação.

Receiver/Transmitter) e interagem com a rede sem fio por meio de conexões TCP/IP. De acordo com a folha de dados do fabricante, as principais características são o uso eficiente de energia, design compacto e desempenho confiável [9].

Ele funciona como uma ponte “serial-wireless” onde são enviados comandos pré-programados no seu *firmware*. Os comandos geralmente são para configuração, inicialização, escrita e leitura dos *buffers* de dados do módulo. Para o uso desta placa de expansão é disponibilizado uma biblioteca que facilita consideravelmente a utilização. Essa biblioteca é mantida por um grupo de usuários e pode ser verificada em *https://github.com/bportaluri/WiFiEsp*.

3.5 IMPLEMENTAÇÃO

Nessa parte será descrito toda a integração da parte de hardware e software referente ao medidor de consumo de energia elétrica.

Dentre os itens relevantes para o desenvolvimento do projeto estão:

- Escolha da entrada analógica referente ao sensor de corrente elétrica.
- Escolha dos pinos para a comunicação serial para o módulo de rede sem fio.
- Definição do período para transmissão de dados para o servidor.
- Definição do padrão de mensagem para envio dos dados para o servidor.
- Definição da verificação do recebimento dos dados pelo servidor.

3.5.1 Hardware

Aproveitando a característica de hardware aberto e extensível da plataforma *Arduino*, praticamente não é necessária nenhuma montagem de circuito

eletrônico. Basicamente as placas de expansões são empilhadas sobre a placa do *Arduino UNO*, conforme Figura 9.

Figura 9: Integração das placas Arduino.[8]

Para a escolha de qual entrada analógica do microcontrolador utilizar, através do diagrama esquemático da placa de expansão apresentado no Anexo B, verifica-se que as entradas A_0, A_1 ou A_2 podem ser utilizadas para esse fim. Essa escolha é importante pois posteriormente será utilizada na programação do software do microcontrolador, ou seja, indicar qual entrada ele deve ler. Para o caso desse projeto será escolhida a entrada A_0.

A comunicação entre o microcontrolador e a placa de expansão de rede sem fio é serial. Para essa comunicação é necessária uma porta de comunicação serial do microcontrolador. No caso do *Arduino UNO* utilizado na implementação, ele possui apenas uma porta de comunicação serial e essa já é utilizada para programação do microcontrolador. Para contornar essa característica uma opção é transformar dois pinos de entrada/saída em uma porta de comunicação serial. Os pinos referentes a I_2 e I_3 serão utilizados sendo I_2 o receptor (R_x) e I_3 o transmissor (T_x).

Na Figura 10 é mostrado o hardware instalado para o monitoramento do consumo de energia elétrica.

[8] Fonte: Fotografia das placas utilizadas na implementação.

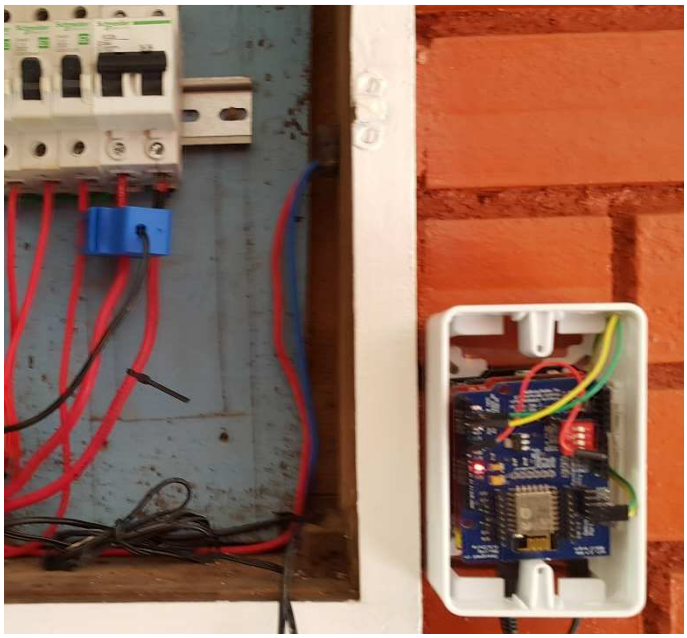


Figura 10: Instalação do medidor de consumo de energia elétrica.

3.5.2 Software

Como visto no capítulo 2, item 2.2.1, o consumo de energia é medido em função da tensão e corrente elétrica, e do tempo. Considerando a tensão de alimentação da rede elétrica como constante (127V) é necessário apenas medir a intensidade da corrente elétrica em intervalos pré-definidos de tempo e totalizar esse valor para obter o consumo de energia elétrica. Foi escolhido arbitrariamente o intervalo de medição de 20 segundos, e o envio do consumo a cada 60 segundos. De forma simplificada esses dois processos podem ser representados pelos fluxogramas na Figura 11.

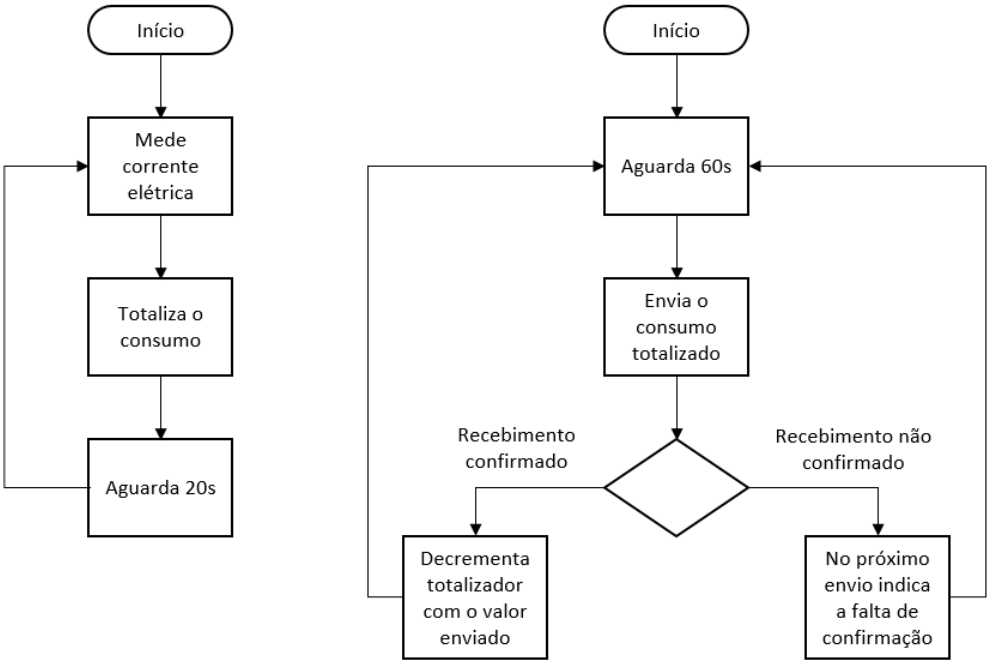


Figura 11: Fluxogramas das tarefas do microcontrolador.

O envio de dados para a Internet é feito baseado no mesmo princípio de funcionamento do sistema *ThingSpeak* [4] utilizado em [3]. De acordo com [4], o *ThingSpeak* é uma plataforma de análise *IoT* que permite agregar, visualizar e analisar fluxos de dados ao vivo na nuvem. Nessa plataforma, os dados são enviados através do método *GET* de requisições *HTTP*, processados e armazenadas em um banco de dados. O Capítulo 4 mostrará a construção de uma versão dedicada dessa plataforma para o projeto, sendo relevante para essa parte apenas o envio da requisição *HTTP*.

Para a mensagem enviada para o servidor foram definidos três parâmetros:

- Consumo no período entre mensagens;
- Quantidade de medições de corrente para o valor do consumo enviado;
- Estado se a última mensagem foi confirmada pelo servidor.

Os dois últimos parâmetros têm como função informar o servidor para uma detecção de falhas na transmissão da mensagem ou da confirmação de

recebimento. A seguir é mostrado um exemplo da requisição *HTTP* do dispositivo para o envio da mensagem ao servidor.

```
enderecodosite.com.br/data.php?consumo=321.54&quant=3&msgant=187
```

A programação da plataforma *Arduino* é feita no ambiente de programação conhecido como *Arduino IDE*, mostrado na Figura 12.

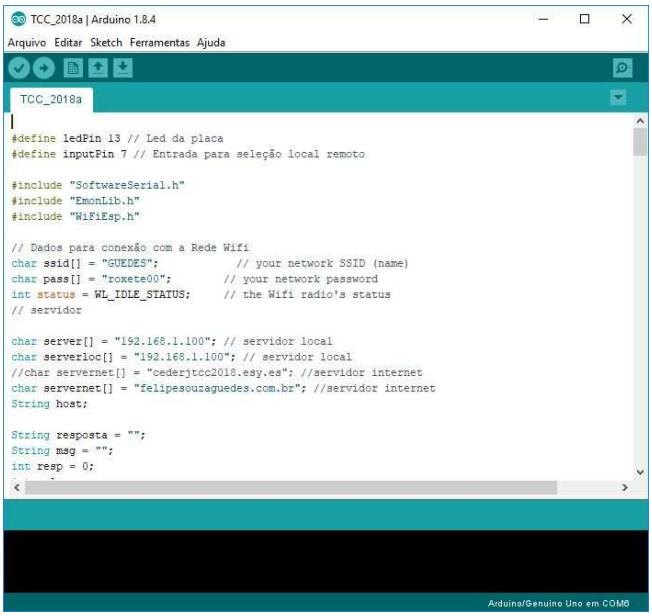


Figura 12: *Arduino IDE*.[9]

Para facilitar a programação pode-se utilizar bibliotecas, mantidas por grupos de usuários. No caso do projeto são usadas três bibliotecas, a primeira devido a placa de expansão para a medição da corrente elétrica, a segunda para a comunicação com módulo de rede sem fio e terceira para criar uma porta serial adicional no dispositivo. A biblioteca para a medição de corrente elétrica é conhecida como *Emonlib*[10]. Para a comunicação com o módulo ESP8266 será usada a biblioteca *WiFiEsp*[11]. A biblioteca *SoftwareSerial* já é normalmente parte do ambiente de programação *Arduino IDE*.

Um código para programação do *Arduino* é composto por cinco blocos principais. O primeiro é composto pela declaração das bibliotecas utilizadas, o

[9] Captura de tela do ambiente de programação do *Arduino*
[10] Biblioteca disponível em *https://github.com/openenergymonitor/EmonLib*
[11] Biblioteca disponível em *https://github.com/bportaluri/WiFiEsp*

segundo pela declaração das variáveis, o terceiro pelas declarações das funções, o quarto pelo bloco *Setup* e, por último o bloco *Loop*.

As funções não precisam ser colocadas abaixo das variáveis, mas são colocadas para se manter uma boa estrutura do código. O código colocado no bloco *Setup* é executado apenas uma única vez na inicialização do dispositivo e é normalmente utilizado para configurar o hardware como os pinos de entrada e saída, a taxa de transmissão da porta serial, entre outras funções. No bloco *Loop* é colocado o programa principal e tem como característica principal a repetição do código quando se chega ao final das instruções.

No Anexo D é mostrado o programa completo do *Arduino*. A seguir é comentado apenas algumas partes mais relevantes do código.

As bibliotecas são incluídas no início do programa através da instrução *include*. Já na parte de definição de variáveis, os itens mais relevantes são as criações das instâncias dos objetos que controlam a medição de corrente e que promovem a comunicação com a rede sem fio. No bloco *Setup*, configura-se a taxa de transmissão de dados das portas seriais, a inicialização do módulo ESP8266 e a conexão com a rede sem fio, a porta onde o sensor de corrente está ligado e a calibração e, a configuração do temporizador. No bloco *Loop* é realizado o cálculo da medição, o envio dos dados e a verificação da resposta do servidor, nos tempos pré-definidos.

4 PROCESSAMENTO DOS DADOS

4.1 SERVIDOR WEB

Para o projeto, o Servidor Web é o responsável por receber os dados provenientes do medidor de consumo e armazená-los em um banco de dados. Outra função é também prover uma página para visualização dos dados colhidos.

O termo Servidor Web refere-se a um computador (*hardware*) que executa um programa (*software*) para a hospedagem de sites na Internet. Ele é o responsável por responder a todas as solicitações feitas para um endereço na Internet.

O *hardware* de um Servidor Web não se difere muito de um computador tradicional, apenas com necessidade de ser mais robusto. Os requisitos mais relevantes para um Servidor Web são sua alta velocidade de processamento, grande memória *RAM* e grande capacidade de armazenamento de dados. Outro item importante relacionado ao Servidor Web, é com relação ao seu endereçamento IP (*Internet Protocol*) que normalmente é do tipo estático, pois o servidor precisa de um endereço fixo para que os servidores *DNS* (*Domain Name System*) sempre encaminhem as requisições para o endereço correto.

No Servidor Web é executado um software cuja função é atender as requisições dos clientes da Web, ou seja, ele fica esperando os dados dos pedidos dos clientes e responde enviando os dados solicitados. Essa interação conhecida como cliente-servidor é o princípio básico de funcionamento da Internet. No Servidor Web também são armazenados todos arquivos associados as páginas como imagens, vídeos, SGBD (Sistemas de Gerenciamento de Banco de Dados), entre outros. Dentre os softwares para um Servidor Web, os mais populares são o *Apache HTTP Server* da *Apache Foundation* e o *IIS* (*Internet Information Services*) da *Microsoft*.

4.2 SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS

De acordo com [10], um Sistema de Gerenciamento de Banco de Dados (SGDB) é uma coleção de programas que permite aos usuários criar e manter um banco de dados. O SGBD é, portanto, um sistema de software de propósito geral que facilita os processos de definição, construção, manipulação e compartilhamento de bancos de dados entre vários usuários e aplicações. A definição de um banco de dados implica especificar os tipos de dados, as estruturas e as restrições para os dados a serem armazenados em um banco de dados.

Os SGBDs trabalham com uma linguagem de programação conhecida como *SQL* (*Structure Query Language* - Linguagem Estruturada de Consulta). Essa linguagem se tornou um padrão para os SGBDs. A *SQL* é uma linguagem de banco de dados abrangente: ela possui comandos para definição de dados, consultas e atualizações.

Existem vários SGBDs no mercado sendo os mais conhecidos o *ORACLE*[12], *MySQL*[13], *PostgreSQL*[14], *SQL-Server*[15], entre outros.

4.3 IMPLEMENTAÇÃO

Nessa parte será descrito o processo de tratamento dos dados enviados pelo medidor de consumo de energia elétrica. Esse processamento consiste em receber o dado, armazená-lo em um banco de dados e disponibilizar esses dados armazenados de uma forma adequada aos usuários. Uma forma bastante intuitiva para apresentação dos dados é a partir de gráficos.

Uma característica bastante importante nessa parte é com relação aos softwares utilizados pelo Servidor Web. Nesse caso, o servidor utilizado será da

[12] Site: https://www.oracle.com/database/index.html
[13] Site: http://www.mysql.com/
[14] Site: http://www.postgresql.org/
[15] Site: http://www.microsoft.com/brasil/servidores/sql/default.mspx

empresa *Hostinger*[16] que disponibiliza gratuitamente a hospedagem de sites e possui a seguinte configuração:

- Suporte a *PHP* (*Apache*)
- Banco de Dados *MySQL*

O Suporte a *PHP* refere-se que o servidor possui instalado o *Apache HTTP Server* da *Apache Foundation*. Normalmente a linguagem de programação para esse tipo de servidor é conhecida como *Linguagem PHP*. A programação do servidor é definida como *Server Side* (“Lado do Servidor”) pois é executada no Servidor Web entregando dados pronto para o cliente. Uma outra característica relevante, para o uso desse software em servidores, é que ele é um software livre, ou seja, não é necessária a aquisição de licença.

O banco de dados *MySQL* é também um software livre que utiliza a linguagem *SQL* (*Structured Query Language*) como interface.

4.3.1 Recebimento e armazenamento dos dados

Antes do recebimento dos dados, primeiramente deverá ser criado e configurado o banco de dados para o armazenamento dos dados. A seguir será mostrado os passos mais importantes para a criação e configuração do banco de dados:

- Criação do banco de dados e do usuário;
- Configuração de outros usuários e dos privilégios para o acesso ao banco de dados;
- Configuração das tabelas que organizaram os dados.

Para a criação do banco de dados, no caso dessa implementação, será utilizado uma ferramenta de administração do site disponibilizado pelo Servidor Web. Essa ferramenta é conhecida como painel de controle que possui muitas outras

[16] Site: *www.hostinger.com.br*

funções além da necessária para esta etapa. Na Figura 13 é apresentado uma tela para a criação do banco de dados.

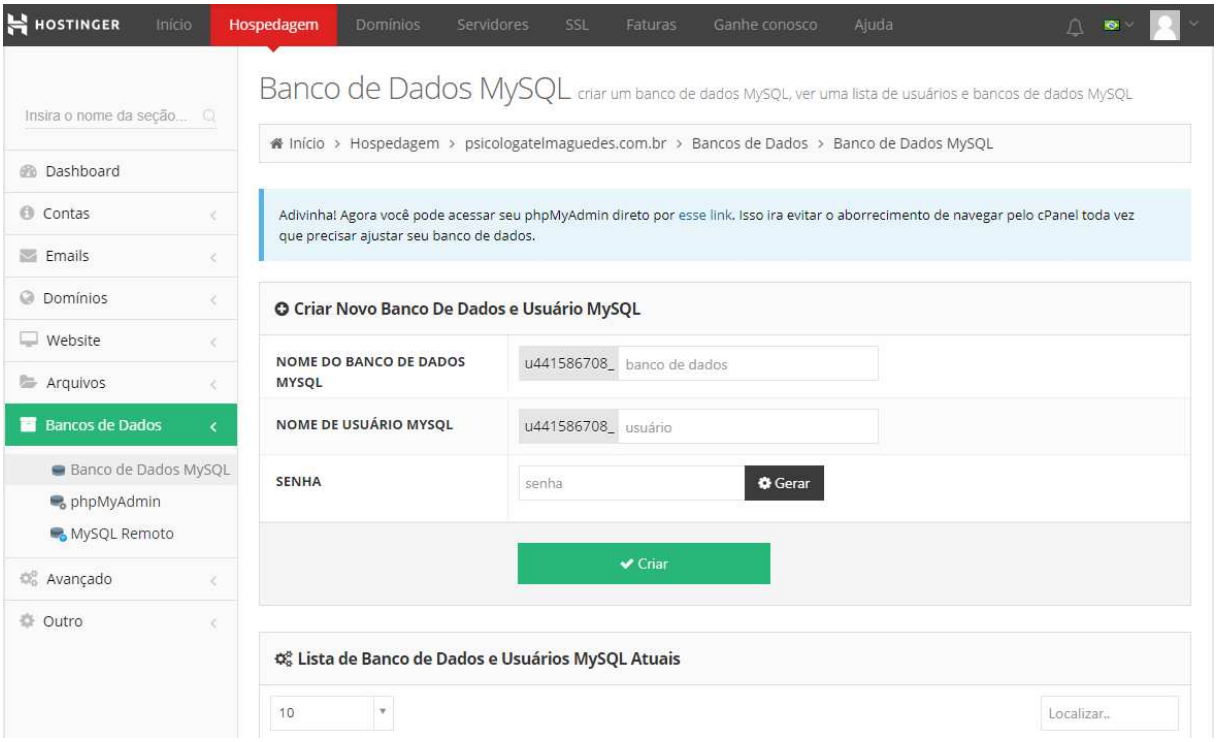


Figura 13: Tela para criação do banco de dados e usuário.

Após a criação do banco, a configuração do mesmo pode ser realizada de várias formas, sendo as mais comuns através de linhas de comandos ou através softwares com interfaces gráficas. O uso de interfaces gráficas é normalmente mais rápido e intuitivo para a configuração. Uma ferramenta para a configuração de bancos de dados *MySQL* é a *phpMyAdmin*[17]. De acordo com [11], o *phpMyAdmin* é um software livre escrito em *PHP*, destinado a lidar com a administração do *MySQL* na *Web*. A interface *phpMyAdmin* é mostrada na Figura 14.

[17]Site: *www.phpmyadmin.net*

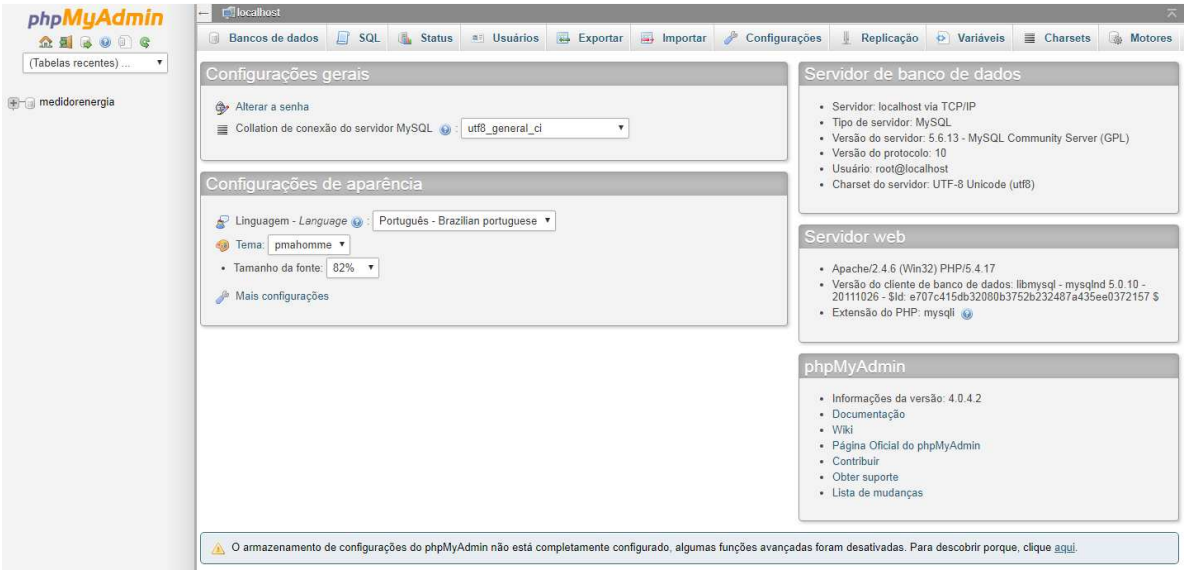


Figura 14: Tela do *phpMyAdmin*.

Utilizando o *phpMyAdmin* será criada a tabela para o armazenamento dos dados enviados pelo medidor de consumo de energia elétrica. Na Tabela 2 é mostrado os campos da tabela.

Tabela 2: Campos da tabela do banco de dados

Campo	**Tipo de dado**	**Descrição**
horario	Datetime	Armazena o horário do envio da informação para o servidor.
consumo	Float	Armazena o valor de consumo enviado pelo medidor.
qtdMedicoes	Int(11)	Armazena a quantidade de medições do consumo no período enviado.
msgAnterior	Int(11)	Armazena o dado referente a confirmação de recebimento do dado pelo servidor.
consumoCor	Float	Armazena o consumo corrigido em função da verificação de recebimento da mensagem anterior.
qtdCor	Int(11)	Armazena a quantidade de medições corrigida em função da verificação de recebimento da mensagem anterior.

Com a estrutura para armazenamento dos dados pronta, é necessário o tratamento dos dados enviados para o servidor via requisição HTTP e escrita no banco de dados.

No ANEXO E é mostrado o código completo implementado no *Servidor Web* para o recebimento e armazenamento dos dados. Na Figura 15 é mostrado um fluxograma para o tratamento dos dados.

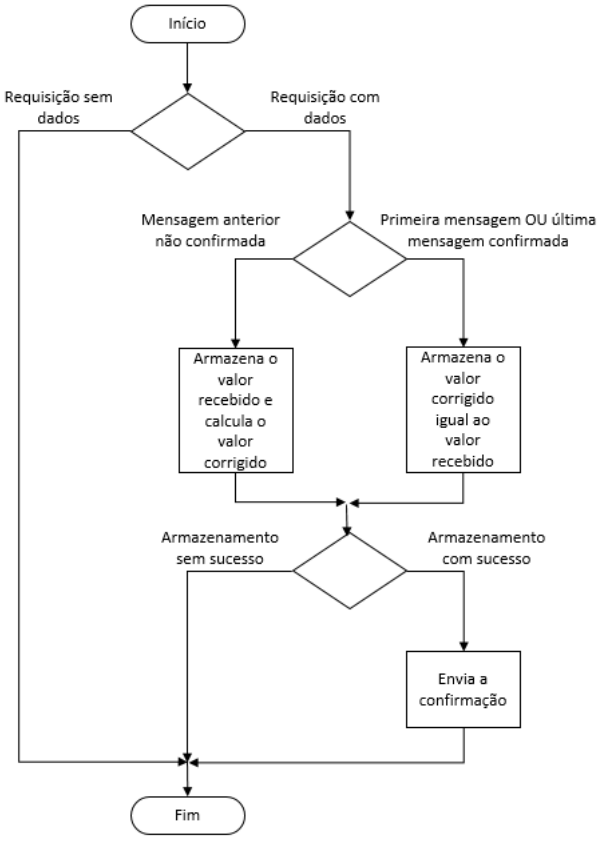


Figura 15: Fluxograma do programa para armazenamento dos dados.

Como visto no item 3.5.2, o medidor emitirá uma requisição *HTTP* para a página que processará os dados. No servidor o primeiro passo é verificar se junto com a requisição foram enviados dados para evitar o processamento caso somente de uma tentativa de acesso a página. Após a confirmação que a requisição possui dados, é verificado se o medidor recebeu uma resposta do servidor referente a transmissão da última mensagem. Se para a última transmissão o medidor recebeu a confirmação do servidor, isso significa que tudo ocorreu perfeitamente. Caso

contrário pode ter ocorrido três problemas: servidor não processou a requisição; servidor processou a requisição, mas a confirmação não chegou ao medidor; ou servidor não conseguiu armazenar a informação no banco de dados. Em todos os casos o servidor não enviará a resposta para o medidor, então este considerará que a última mensagem não obteve sucesso e indicará essa situação na próxima mensagem. Com a informação da falha da confirmação da última mensagem, o servidor fará uma consulta no banco de dados para verificar o último dado salvo, e comparando o número de medições, concluirá se a última mensagem foi processado ou não. Com base nessa condição o sistema fará a correção da medição para o armazenamento.

4.3.2 Disponibilização dos dados

Os dados coletados pelo medidor e armazenados no servidor poderão ser consultados através de um acesso a uma página WEB destinada para esse fim. Para o usuário a informação do consumo é mais intuitiva e interessante utilizando-se um gráfico ao invés de uma tabela com as medições. A taxa de envio dos dados de consumo é de um envio por minutos. Essa taxa para exibição de um dia de monitoração é muito alta então é mais adequado agrupar esse dado numa faixa maior. Para o projeto a visualização será escolhida arbitrariamente de períodos de quinze minutos.

Uma ferramenta para a construção de gráfico para a visualização dos dados é uma *API* da empresa *Google*, conhecida como *Google Chart*[18]. Essa API funciona simplificando todo o processo gráfico, sendo necessário apenas carregar a tabela de dados na página de destino. Essa *API* utiliza os recursos do lado do cliente (*Client Side*) construindo o gráfico utilizando os recursos de *Javascript* existente nos navegadores de Internet.

O código da página para visualização dos dados de consumo é mostrado no ANEXO F e a página Web na Figura 16.

[18] https://developers.google.com/chart/

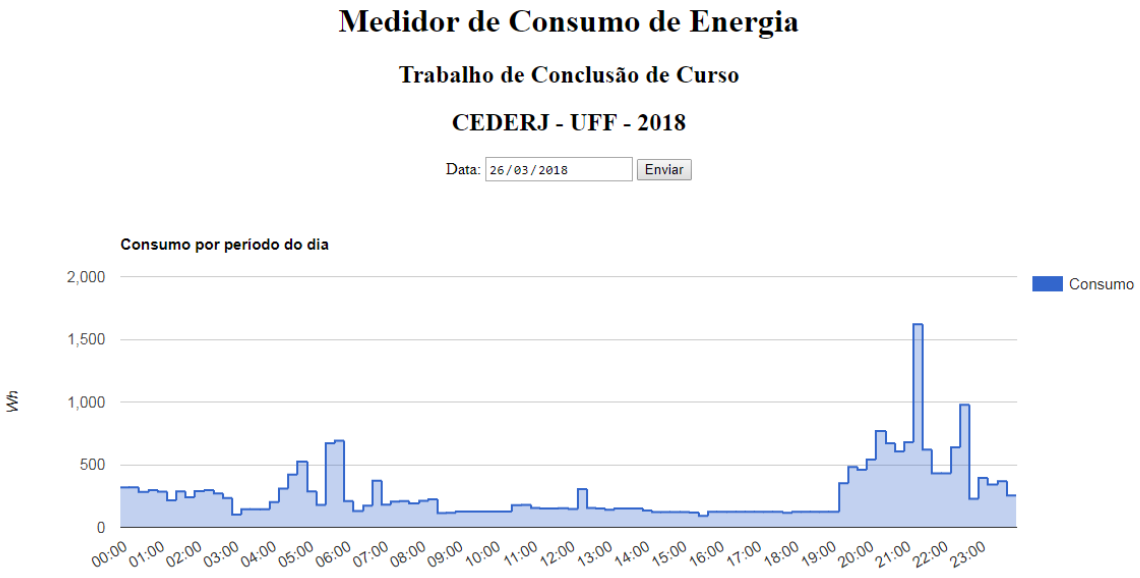


Figura 16: Página Web para disponibilização dos dados.

Na página o usuário apenas seleciona a data que ele deseja visualizar, e será mostrada a página.

5 RESULTADOS E ANÁLISE

5.1 RESULTADOS DA IMPLEMENTAÇÃO

Na implementação os fatores mais importantes foram a modularidade da plataforma de *hardware* e as bibliotecas de *softwares* que possuem uma ótima reusabilidade devido ao seu encapsulamento. A reusabilidade reduz os custos de produção e manutenção. O encapsulamento faz com detalhes internos das classes sejam desnecessários do ponto de vista do projeto.

No caso da implementação do medidor, as bibliotecas *Emonlib*, *WiFiEsp* e *SoftwareSerial* utilizadas na plataforma *Arduino* são de utilização bastante simplificadas não necessitando de um aprofundamento no funcionamento dos itens de hardware.

A configuração do SGBD *MySQL* é bastante facilitada com uso da interface *phpMyAdmin*, pois mostra as opções dos parâmetros e também dispensa o uso de comandos SQL para a configuração das tabelas e acessos.

Para a disponibilização dos dados a *API Google Chart* também facilita o trabalho para visualização dos dados, apenas necessitando a entrada dos dados para a construção do gráfico.

5.2 ANÁLISE DO CONSUMO DA RESIDÊNCIA

Para a verificação da aplicabilidade do projeto foram escolhidas datas para a análise com consumo diário e uma análise do consumo mensal de uma residência.

As datas para a análise diária foram 03, 04, 07, 08, 13, 16 e 17 de abril de 2018 e o referido mês para a análise mensal.

A Figura 17 mostra o consumo da residência no dia 03 de abril de 2018.

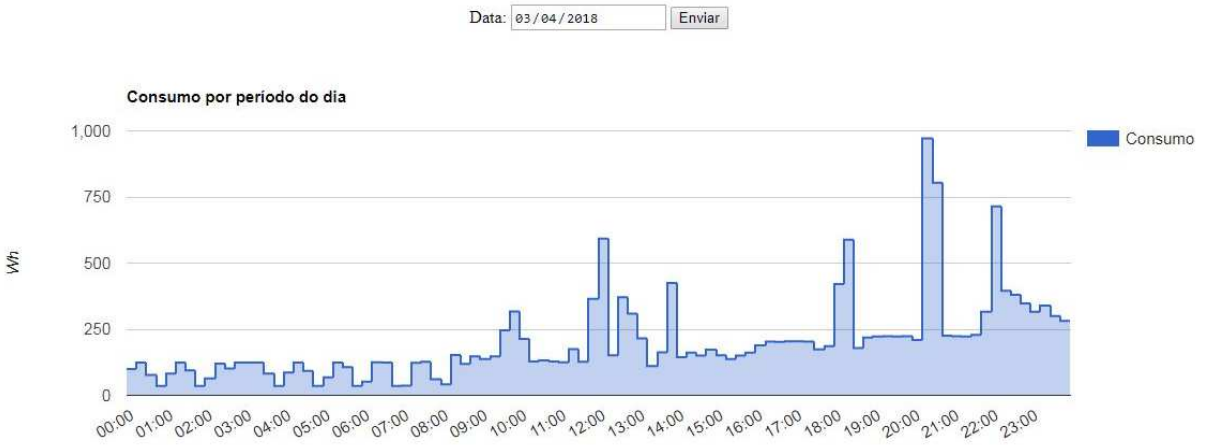


Figura 17: Consumo do dia 03/04/2018.

Nesse dia os moradores chegaram de viagem por volta das 08h30, quando se verifica o aumento do consumo de energia da residência. No período das 12h00 às 13h00 são ligados os equipamentos (micro-ondas e fritadeira elétrica) para o preparo das refeições do almoço. Já no período da tarde (após 13h00) verifica-se um aumento médio do consumo provocado pelo funcionamento de uma lavadora de roupas. Às 18h00, 20h30 e 22h30 aconteceram picos no consumo devido ao uso do chuveiro elétrico para os banhos. Após às 22h00 o consumo também aumenta devido ao uso dos aparelhos de ar condicionado.

A Figura 18 mostra o consumo da residência no dia 04 de abril de 2018.

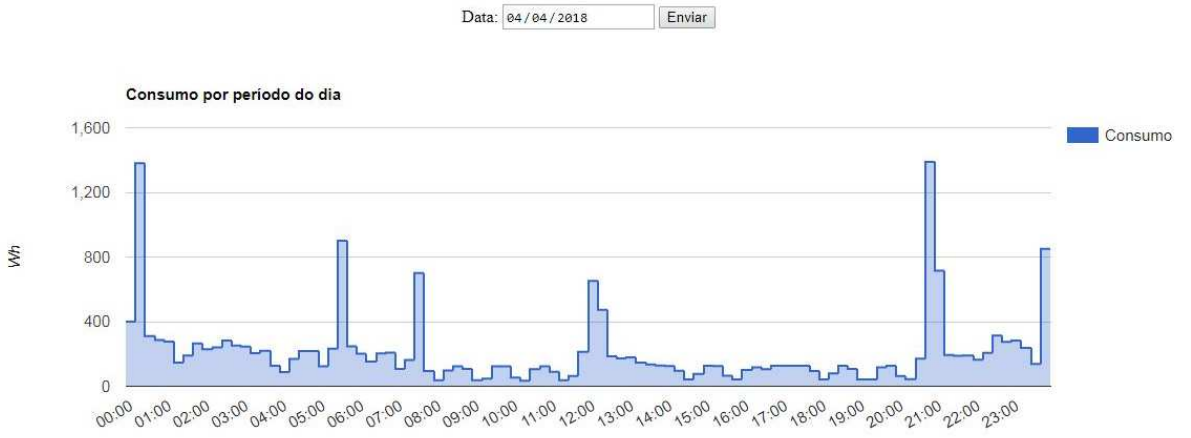


Figura 18: Consumo do dia 04/04/2018.

Por volta das 05h30 e 07h30, ocorrem um aumento do consumo devido ao uso do chuveiro elétrico para o banho. Às 12h00 acontece um aumento pelo uso

do forno micro-ondas. Às 21h00 é novamente utilizado o chuveiro elétrico e ligado os aparelhos de ar condicionado aumentando o consumo.

A Figura 19 mostra o consumo da residência no dia 07 de abril de 2018.

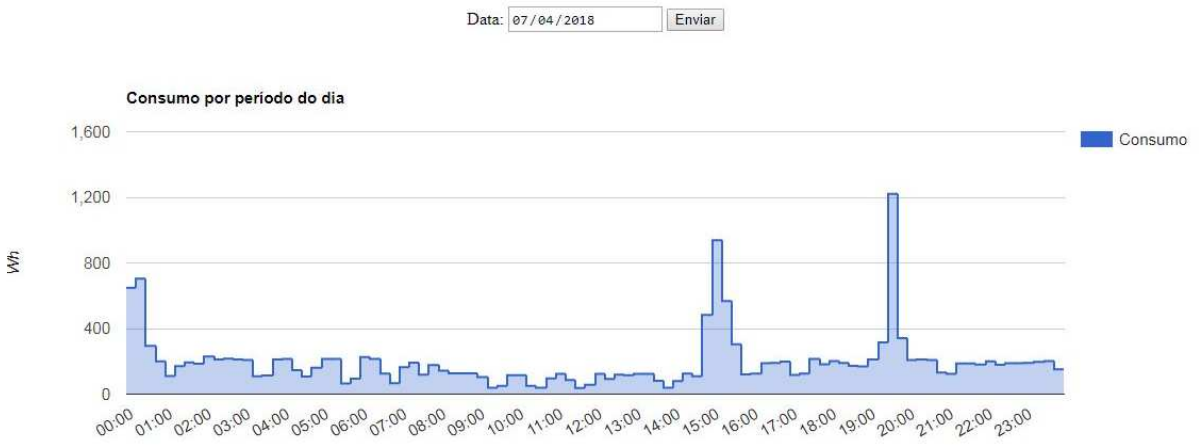


Figura 19: Consumo do dia 07/04/2018.

Nesta data os moradores não ficaram em casa na parte da manhã, apenas retornando por voltas das 14h30, onde o consumo foi aumentado devido ao uso dos aparelhos para preparo do almoço e chuveiro elétrico para banho.

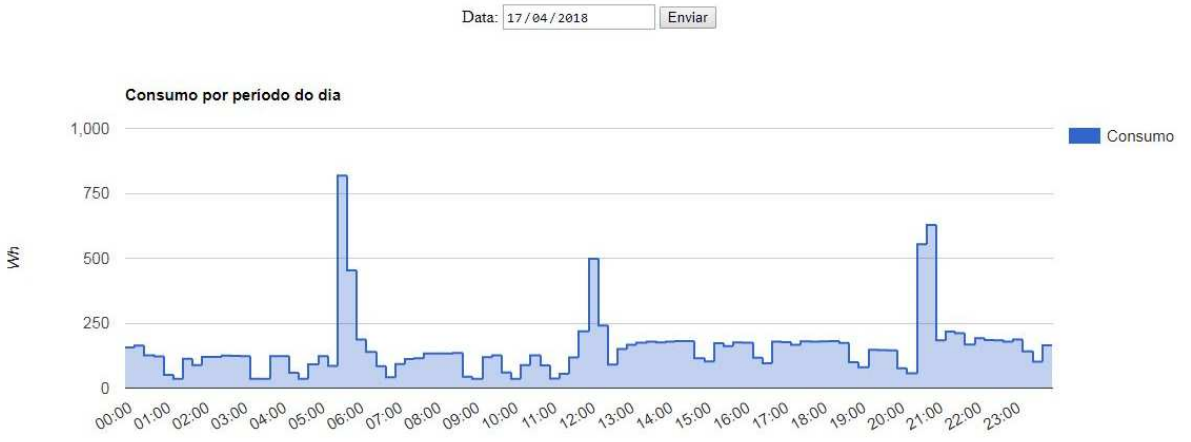


Figura 20: Consumo do dia 17/04/2018

A partir dos dados registrado no banco de dados pode-se criar um gráfico (Figura 21) para a análise do consumo mensal.

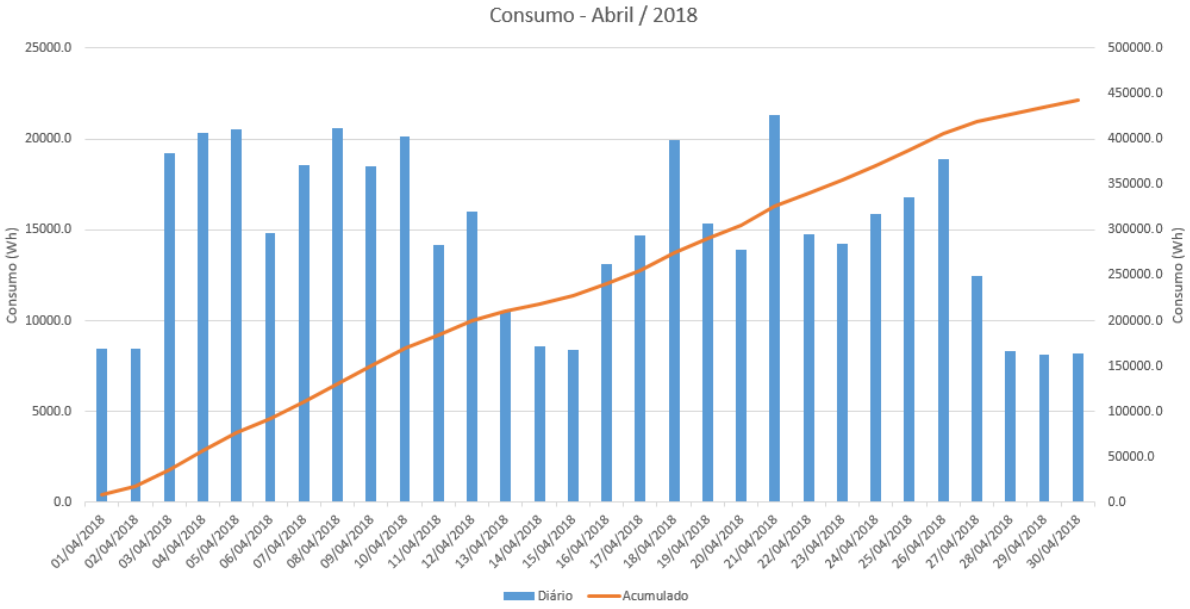


Figura 21: Consumo Abril/2018

Com base no gráfico pode ser verificado, o menor consumo nos três períodos onde os moradores estavam ausentes da residência: entre os dias 1 e 2, 14 e 15, e 28 a 30 de abril.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho mostrou mais um item conectado à Internet além de computadores ou *smartphones*. A Internet das Coisas cada vez mais presente no dia-a-dia das pessoas auxilia a obter informações com mais qualidade e facilidade.

Essa implementação mostrou um acompanhamento aceitável do consumo de uma residência com uma resolução bastante interessante. Antes o dado de consumo só era conhecido no final do mês através da conta de energia da concessionária sem nenhum detalhe de horário de maior consumo. Com um medidor conectado a Internet, o usuário poderia verificar o consumo de qualquer lugar simplesmente acessando uma página Web através de um computador ou *smartphone*.

As tecnologias disponíveis tanto de *hardware* como de *software* possuem uma reusabilidade muito alta, o que faz com sua integração seja rápida sem o desprendimento de muitos recursos ou conhecimentos aprofundados. A plataforma *Arduino* aberta, de baixo custo e com uma vasta quantidade de material disponível fez com a implementação do *hardware* fosse facilitada. Os itens de *software* utilizados foram também de plataforma aberta não gerando custo de desenvolvimento não inviabilizando a implementação.

Durante as pesquisas e estudos dos itens necessários, foram encontrados inúmeras aplicações e soluções para os mais diferentes problemas o que mostra que o conceito de Internet das Coisas é muito amplo e pode ser ainda muito explorado.

Esse trabalho ainda pode servir de base para outras ideias, assim como possui muitos pontos de melhorias e aplicação de outros conceitos, como armazenagem dos dados em cartões de memória no caso de uma indisponibilidade da rede, conexão da Internet por meio de redes de celulares, emissão de relatórios por e-mail, entre outros.

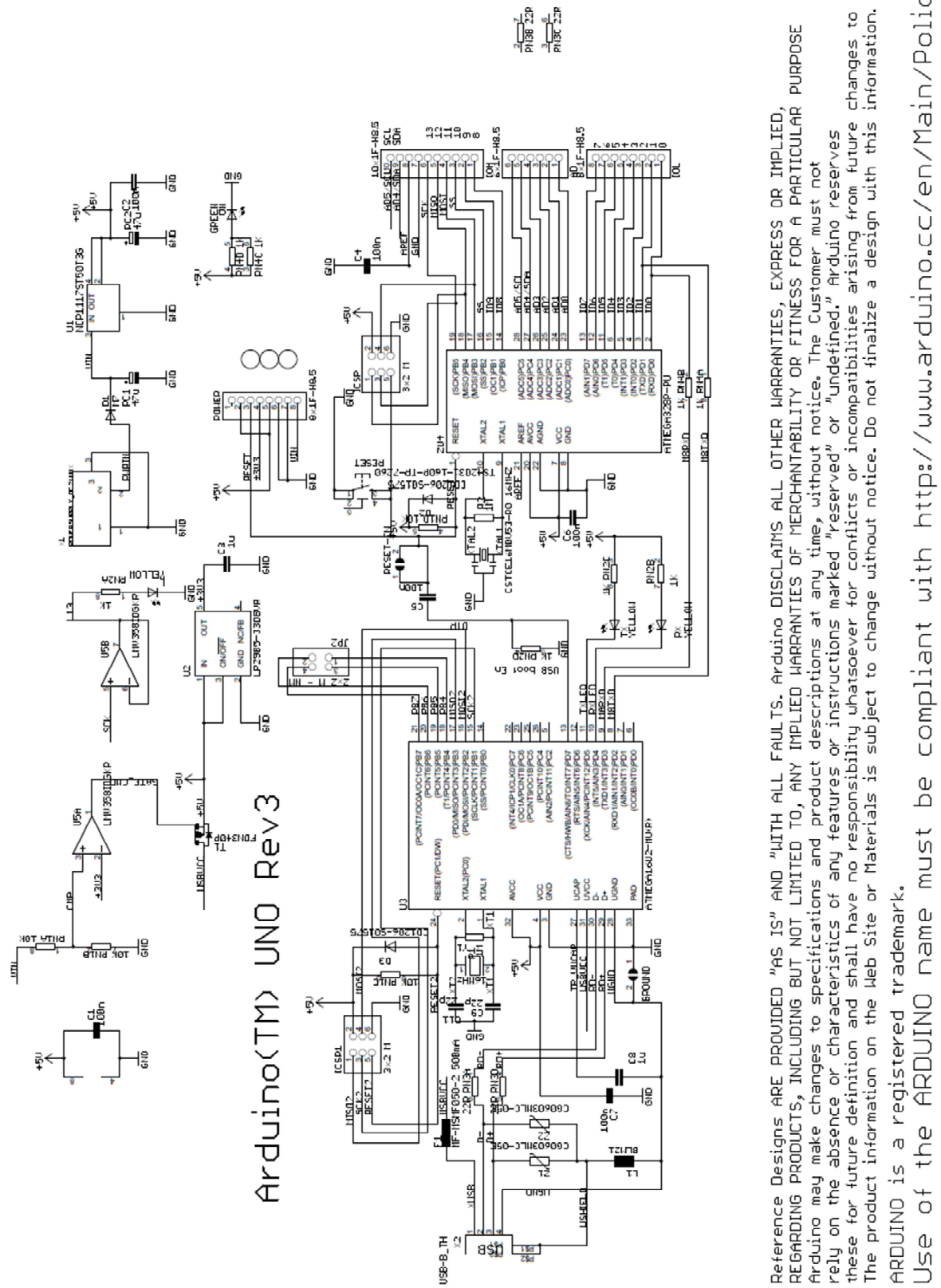
REFERÊNCIAS BIBLIOGRÁFICAS

1. MONK, S. **Programação com Arduino II:** passos avançados com sketches / Simon Monk; tradução: Anatólio Laschuk. Porto Alegre: Bookman, 2015.
2. ZAMBARDA, P. 'Internet das Coisas': entenda o conceito e o que muda com a tecnologia. **TechTudo**, 16 ago. 2014. Disponivel em: <http://www.techtudo.com.br/noticias/noticia/2014/08/internet-das-coisas-entenda-o-conceito-e-o-que-muda-com-tecnologia.html>. Acesso em: 9 Agosto 2017.
3. FERNANDES, A. C. et al. Sistema de aquisição de sinais ECG processado pelo LabVIEW com comunicação wi-fi por meio do módulo ESP8266. **Revista Principia**, 01 Junho 2017. 62-68.
4. THE MATHWORKS, INC. IoT Analytics - ThingSpeak Internet of Things. Disponivel em: <https://thingspeak.com/>. Acesso em: 07 mar. 2018.
5. OPENENERGYMONITOR PROJECT. OpenEnergyMonitor project. Disponivel em: <https://openenergymonitor.org/>. Acesso em: 05 Janeiro 2018.
6. ARDUINO AG. Arduino - Introduction. Disponivel em: <https://www.arduino.cc/en/Guide/Introduction#>. Acesso em: 28 set. 2017.
7. ATMEL CORPORATION. ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES IN-SYSTEM PROGRAMMABLE FLASH DATASHEET, 2015. Disponivel em: <http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf>. Acesso em: 28 dez. 2017.
8. ARDUINO AG. Arduino - Arduino Uno WiFi. Disponivel em: <https://www.arduino.cc/en/Guide/ArduinoUnoWiFi>. Acesso em: 27 fev. 2018.
9. ESPRESSIF INC. ESP8266EX Datasheet. Disponivel em: <https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf>. Acesso em: 27 fev. 2018.
10. RAMEZ, E.; NAVATHE, S. B. **Sistemas de banco de dados**. São Paulo: Pearson Addison Wesley, 2005.

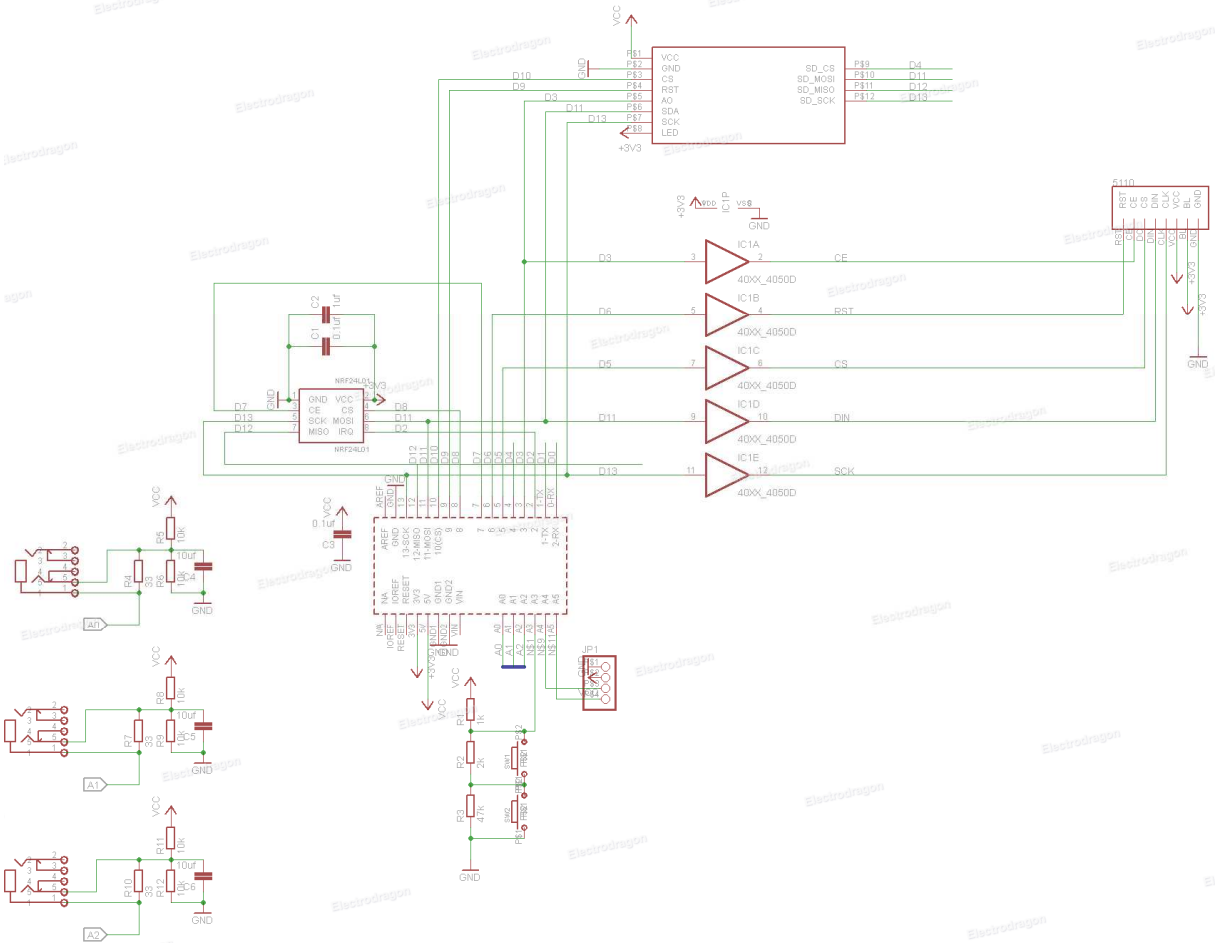
11. PHPMYADMIN CONTRIBUTORS. Bringing MySQL to the web. Disponivel em: <https://www.phpmyadmin.net/>. Acesso em: 20 mar. 2018.
12. MONK, S. **Programação com Arduino:** começando com Sketches / Simon Monk; tradução Anatólio Laschuk. Porto Alegre: Bookman, 2013.

ANEXOS

ANEXO A - Diagrama esquemático *Arduino UNO*.



ANEXO B - Diagrama esquemático da placa de expansão para ligação dos sensores de corrente.



ANEXO C - Folha de dados do sensor de corrente.

Split core current transformer

Model: SCT-013 **Rated input current: 5A/100A**

Characteristics: Openning size: 13mm*13mm,
Non-linerity±3% (10%—120% of rated input current)
1m leading wire, standard Φ3.5 three core plug output.
Current output type and voltage output type (voltage output type built-in sampling resistor)

Purpose: Used for current measurement, monitor and protection for AC motor, lighting equipment, air compressor etc

Core material: ferrite

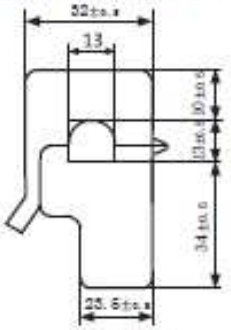
Mechanical strength: the number of switching is not less than 1000 times(test at 25℃)

Safty index: Dielectric strength(between shell and output)1000V AC/1min

Fire resistance property: In accordance with UL94-Vo

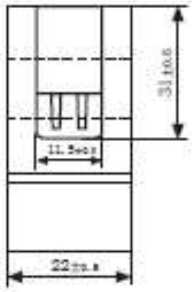
Work temperature: -25℃~+70℃

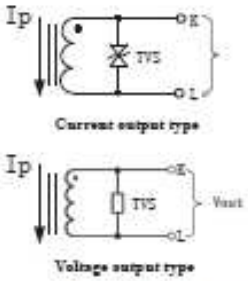
Outline size diagram: (in mm)



Front View

Side View





Schematic diagram

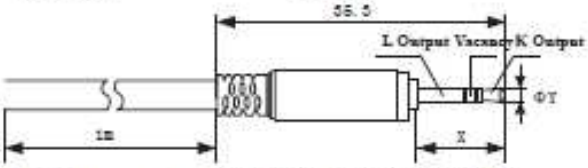


Diagram for standard three-core plug

Three-core plug size

	X	Y	
2.5mm Audio Plug	11.8	2.5	Optional
3.5mm Audio Plug	15.0	3.5	standard

Table of technical parameter:

Model	SCT-013-000	SCT-013-005	SCT-013-010	SCT-013-015	SCT-013-020
Input current	0-100A	0-5A	0-10A	0-15A	0-20A
Output type	0-50mA	0-1V	0-1V	0-1V	0-1V
Model	SCT-013-025	SCT-013-030	SCT-013-050	SCT-013-060	SCT-013-000V
Input current	0-25A	0-30A	0-50A	0-60A	0-100A
Output type	0-1V	0-1V	0-1V	0-1V	0-1V

※ Output type: voltage output type built-in sampling resistor, current output type built-in protective diode.

ANEXO D - Código implementado no medidor de energia.

```
#define ledPin 13 // Led da placa
#define inputPin 7 // Entrada para seleção local remoto

#include "SoftwareSerial.h"
#include "EmonLib.h"
#include "WiFiEsp.h"

// Dados para conexão com a Rede Wifi
char ssid[] = "GUEDES";            // your network SSID (name)
char pass[] = "roxete00";        // your network password
int status = WL_IDLE_STATUS;     // the Wifi radio's status
// servidor

char servernet[] = "felipesouzaguedes.com.br"; //servidor internet
String host;

String resposta = "";
String msg = "";
int resp = 0;
int val;

SoftwareSerial Serial1(3, 2); // RX, TX
EnergyMonitor emon1;
WiFiEspClient client;

int tempoTimer=0x85EE; // // 65536-(16MHz/1024/0,5Hz) = 0x85EE para
2s
int contadorTempo1=0; // variavel que será incrementada a cada
contagem
int executaTempo1=0; // armadilha para executar
int contadorTempo2=0; // variavel que será incrementada a cada
contagem
int executaTempo2=0; // armadilha para executar
```

```
double Irms1;
double Pot1;
double Consumo;

double totalizadorConsumo=0; //totalizar do consumo
double totalizadorParaTx=0;
int q1=0;
int q1Tx;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(inputPin, INPUT);
  delay(5000); // atraso para inicializaÃ§Ã£o

//************************************************************
//Inicialização Portas Seriais
//************************************************************
  Serial.begin(9600);  // initialize serial for Debug
  Serial1.begin(9600);  // initialize serial for ESP module
//************************************************************



//************************************************************
//Inicialização do Módulo Rede sem fio
//************************************************************

  WiFi.init(&Serial1); // initialize ESP module

// check for the presence of the shield
  if (WiFi.status() == WL_NO_SHIELD)
  {
    Serial.println("WiFi shield not present");
    // don't continue
    while (true);
  }
```

```
  // attempt to connect to WiFi network
  while ( status != WL_CONNECTED)
  {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network
    status = WiFi.begin(ssid, pass);
  }

  // you're connected now, so print out the data
  Serial.println("You're connected to the network");
  printWifiStatus();
  Serial.println();
//************************************************************


//************************************************************
//Inicialização leitura de corrente
//************************************************************
  emon1.current(0, 111.1);             // Current: input pin,
calibration.

//************************************************************


//************************************************************
//Inicialização do Temporizador (Timer1)
//************************************************************
  // Configuração do timer1
  TCCR1A = 0;                          // confirma timer para operação
normal pinos OC1A e OC1B desconectados
  TCCR1B = 0;                          // limpa registrador
  TCCR1B |= (1<<CS10)|(1 << CS12);   // configura prescaler para
1024: CS12 = 1 e CS10 = 1
  TCNT1 = tempoTimer;
```

```
  TIMSK1 |= (1 << TOIE1);            // habilita a interrupção do
TIMER1
//************************************************************

}

//************************************************************
//Loop
//************************************************************
void loop()
{
    host = "Host: felipesouzaguedes.com.br";

  // Bloco para medição da corrente e cálculo do consumo em 20s
  if (executaTempo1==1)
  {
    executaTempo1=0;
  // faz as medições de correntes
    Irms1 = emon1.calcIrms(1480);  // Calculate Irms only
  // calcula a potência
    Pot1 = Irms1*127;

// calcula o consumo
    Consumo = Pot1*20.00/3600.00;
    totalizadorConsumo = totalizadorConsumo + Consumo; //
totalizando o consumo
    q1 = q1+1;

         printDados();

   }

// Bloco para transferir os dados de consumo
  if (executaTempo2==1)
  {
    executaTempo2=0;
```

```
     //código para x tempo
totalizadorParaTx = totalizadorConsumo;
q1Tx = q1;
     //Serial.println("Transferir Dados");

     msg="";
msg.concat("GET /data.php?consumo=");
msg.concat(String(totalizadorParaTx, 8));
msg.concat("&quant=");
msg.concat(q1Tx);
msg.concat("&msgant=");
msg.concat(resp);
msg.concat(" HTTP/1.1");

//Serial.println(msg);
//Serial.println(host);

     // if you get a connection, report back via serial
     if (client.connect(server, 80))
{
               sendRequest();
     }

// if there are incoming bytes available
// from the server, read them and print them

resposta = "";
while (client.available())
{
  char c = client.read();
  resposta.concat(c);
  Serial.write(c); //remover
}
//Serial.println(); //remover
client.flush();
client.stop();
```

```
    resp = resposta.indexOf("registro=1");

         // se houve resposta
    if (resp!=-1)
    {
      totalizadorConsumo=totalizadorConsumo-totalizadorParaTx;
      q1 = q1-q1Tx;
    }
  }
}

//*************************************************************
//Função para mostrar o status da conexão com a rede sem fio
//*************************************************************
void printWifiStatus()
{
  // print the SSID of the network you're attached to
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi shield's IP address
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print the received signal strength
  long rssi = WiFi.RSSI();
  Serial.print("Signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}
//*************************************************************
//Função para mandar requisição HTTP
//*************************************************************
void sendRequest()
```

```
{
  //Serial.println("Connected to server");
  // Make a HTTP request
  client.println(msg);
  client.println(host);
  client.println("Connection: close");
  client.println();
}
//*************************************************************
//Imprime os dados na serial
//*************************************************************
void printDados()
{
    //Serial.print("Irms1: ");
    Serial.print(Irms1);
    //Serial.println(" A ");
    Serial.print(" / ");

    //Serial.print("Consumo: ");
    Serial.print(Consumo);
    //Serial.println(" Wh ");
    Serial.print(" / ");

    //Serial.print("Totalizador Consumo: ");
    Serial.print(totalizadorConsumo);
    //Serial.println(" Wh ");
    Serial.print(" / ");

    Serial.print(q1);

    Serial.println("");
}
//*************************************************************
//Interrupção do Temporizador
//*************************************************************
ISR(TIMER1_OVF_vect) //interrupção do TIMER1
```

```
{
  TCNT1 = tempoTimer; //Renicia TIMER

  digitalWrite(ledPin, digitalRead(ledPin) ^ 1); //inverte estado do
led

  // Tempo para tarefa 1 (20s)
  contadorTempo1++;
  if (contadorTempo1==10)
  {
    contadorTempo1 = 0;
    executaTempo1 = 1;
  }

  // Tempo para tarefa 2  (60s)
  contadorTempo2++;
  if (contadorTempo2==30)
  {
    contadorTempo2 = 0;
    executaTempo2 = 1;
  }
}
```

ANEXO E - Código implementado para recebimento e armazenamento.

```
<?php
if
(isset($_GET["consumo"])&&isset($_GET["quant"])&&isset($_GET["msgant
"]))
{
        $consumo = floatval($_GET["consumo"]);
        $quantidade = intval($_GET["quant"]);
        $msganterior = intval($_GET["msgant"]);

        date_default_timezone_set('America/Sao_Paulo');
        $mysqltime = date ("Y-m-d H:i:s", strtotime("now"));

        $con = mysql_connect("mysql.hostinger.com.br",
"u210277023_usr01", "roxete") or die ("Sem conexão com o servidor");
        $select = mysql_select_db("u210277023_db01") or die("Sem
acesso ao DB");
        //$con = mysql_connect("localhost", "root", "usbw") or die
("Sem conexão com o servidor");
        //$select = mysql_select_db("medidorenergia") or die("Sem
acesso ao DB");


        if ($msganterior >= 0)
        {
                $consumoCor = $consumo;
                $quantidadeCor = $quantidade;
        }
        else
        {
                $cmd = "SELECT * FROM tblconsumo
                                WHERE horario = (SELECT
MAX(horario) FROM tblconsumo);";
                $result = mysql_query($cmd);
```

```
                $anterior = mysql_fetch_assoc($result);
                echo $quantidade;
                echo $anterior["qtdMedicoes"];
                if($quantidade<=$anterior["qtdMedicoes"])
                {
                        $consumoCor = $consumo;
                        $quantidadeCor = $quantidade;
                }
                else
                {
                        $consumoCor = $consumo -
$anterior["consumo"];
                        $quantidadeCor = $quantidade -
$anterior["qtdMedicoes"];
                }
        }

        $cmd = "INSERT INTO tblconsumo (horario, consumo,
qtdMedicoes, msgAnterior, consumoCor, qtdCor)
                        VALUES ('".$mysqltime."',".$consumo.",
".$quantidade.",  ".$msganterior.",  ".$consumoCor.",
".$quantidadeCor.");";

        $result = mysql_query($cmd);

        if ($result) {
                echo "registro=1";
        }
}
?>
```

ANEXO F - Código implementado para disponibilizar os dados.

```
<?php

date_default_timezone_set('America/Sao_Paulo');

if (isset($_POST["data"]))
{
        $dataInput = $_POST["data"];
}
else
{
        $dataInput = date ("Y-m-d", strtotime("now - 1 days"));
}

$dataEscolhidaInicial = date_create($dataInput);
$dataEscolhidaFinal = date_create($dataInput);
date_add($dataEscolhidaFinal,
date_interval_create_from_date_string('1 days'));

$con = mysql_connect("mysql.hostinger.com.br", "u210277023_usr01",
"roxete") or die ("Sem conexão com o servidor");
$select = mysql_select_db("u210277023_db01") or die("Sem acesso ao
DB");

//$con = mysql_connect("localhost", "root", "usbw") or die ("Sem
conexão com o servidor");
//$select = mysql_select_db("medidorenergia") or die("Sem acesso ao
DB");

$cmd = "SELECT * FROM tblconsumo
                WHERE horario >
'".date_format($dataEscolhidaInicial, 'Y-m-d H:i:s')."'
                AND horario < '".date_format($dataEscolhidaFinal,
'Y-m-d H:i:s')."';";

                $result = mysql_query($cmd);
```

```
?>

<html>
  <head>
    <script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
      google.charts.load('current', {'packages':['corechart']});
      google.charts.setOnLoadCallback(drawChart_1);
           google.charts.setOnLoadCallback(drawChart_2);

      function drawChart_1() {
        var data = google.visualization.arrayToDataTable([
          ['Horário',  'Consumo']

<?php
$consumo = 0;

$horaInicial = date_create($dataInput);
$horaFinal = date_create($dataInput);
date_add($horaFinal, date_interval_create_from_date_string('15
minutes'));

while($registro = mysql_fetch_assoc($result))
{
        $hora = DateTime::createFromFormat('Y-m-d H:i:s',
$registro['horario']);
        if ($hora<$horaFinal)
        {
                $consumo = $consumo + $registro['consumoCor'];
        }
        else
        {
```

```
                printf(",['%s', %f ]", date_format($horaInicial,
'H:i'), $consumo);
                $consumo = $registro['consumoCor'];
                date_add($horaInicial,
date_interval_create_from_date_string('15 minutes'));
                date_add($horaFinal,
date_interval_create_from_date_string('15 minutes'));
        }
}
                printf(",['%s', %f ]", date_format($horaInicial,
'H:i'), $consumo);
?>
        ]);
        var options = {
          title: 'Consumo por período do dia',
          vAxis: {title: 'Wh'},
          isStacked: true
        };
        var chart = new
google.visualization.SteppedAreaChart(document.getElementById('chart
_1_div'));
        chart.draw(data, options);
      }

      function drawChart_2() {
        var data = google.visualization.arrayToDataTable([
          ['Horário',  'Consumo'],
          ['00:00 - 01:00', 1],
          ['01:00 - 02:00', 2],
          ['02:00 - 03:00', 3],
                  ['03:00 - 04:00', 2],
                  ['04:00 - 05:00', 1],
                  ['05:00 - 06:00', 2],
                  ['06:00 - 07:00', 3],
                  ['07:00 - 08:00', 2],
                  ['08:00 - 09:00', 1],
```

```
                    ['09:00 - 10:00', 2],
                    ['10:00 - 11:00', 3],
                    ['11:00 - 12:00', 2],
                    ['12:00 - 13:00', 1],
                    ['13:00 - 14:00', 2],
                    ['14:00 - 15:00', 3],
                    ['15:00 - 16:00', 2],
                    ['16:00 - 17:00', 1],
                    ['17:00 - 18:00', 2],
                    ['18:00 - 19:00', 3],
                    ['19:00 - 20:00', 2],
                    ['20:00 - 21:00', 1],
                    ['21:00 - 22:00', 2],
                    ['22:00 - 23:00', 3],
                    ['23:00 - 00:00', 2]
        ]);
        var options = {
          title: 'Consumo por período do dia',
          vAxis: {title: 'Wh'},
          isStacked: true
        };
        var chart = new
google.visualization.SteppedAreaChart(document.getElementById('chart
_2_div'));
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
        <div align="center">
                <h1>Medidor de Consumo de Energia</h1>
                <h2>Trabalho de Conclusão de Curso</h2>
                <h2>CEDERJ - UFF - 2018</h2>
                <form action="grafico.php" method="post">
                  Data:
```

```
                    <input name="data" type="date" value=<?php echo
$dataInput;?>>
                    <input type="submit">
                  </form>

                  <div id="chart_1_div" style="width: 1200px; height:
400px;"></div>
                  <!--<div id="chart_2_div" style="width: 1200px;
height: 400px;"></div>-->
          </div>
  </body>
</html>
```