UNIVERSIDADE ESTADUAL DE MATO GROSSO DO SUL

UNIDADE UNIVERSITÁRIA DE NOVA ANDRADINA

CURSO DE COMPUTAÇÃO, LICENCIATURA

FUTURO SOLAR: UM SERVIDOR AUTÔNOMO ALIMENTADO POR ENERGIA SOLAR

SILVANO CRIVELLI DA SILVA

NOVA ANDRADINA - MS

2018

UNIVERSIDADE ESTADUAL DE MATO GROSSO DO SUL

UNIDADE UNIVERSITÁRIA DE NOVA ANDRADINA

CURSO DE COMPUTAÇÃO, LICENCIATURA

FUTURO SOLAR: UM SERVIDOR AUTÔNOMO ALIMENTADO POR ENERGIA SOLAR

Trabalho de Conclusão de Curso apresentado ao Curso de Computação, Licenciatura, da Universidade Estadual de Mato Grosso do Sul – Unidade de Nova Andradina, como requisito parcial para a obtenção do título de Licenciado em Computação.
Orientador: Prof. Esp. André Castro Garcia

SILVANO CRIVELLI DA SILVA

NOVA ANDRADINA - MS

2018

S583f Silva, Silvano Crivelli

Futuro solar: um servidor autônomo alimentado por energia solar/ Silvano Crivelli da Silva. Nova Andradina, MS: UEMS, 2018.

71p. ; 30cm.

Monografia (Especialização) – Computação, Licenciatura – Universidade Estadual de Mato Grosso do Sul, 2018.

Orientador: Prof. Esp. André Castro Garcia.

1. Arduino 2. Automação residencial 3. Energia solar 4. Renovável I. Título.

CDD 23.ed. 621.31244

SILVANO CRIVELLI DA SILVA

FUTURO SOLAR: UM SERVIDOR AUTÔNOMO ALIMENTADO POR ENERGIA SOLAR

Trabalho de Conclusão de Curso apresentado ao Curso de Computação, Licenciatura, da Universidade Estadual de Mato Grosso do Sul – Unidade de Nova Andradina, como requisito parcial para a obtenção do título de Licenciado em Computação.
Orientador: Prof. Esp. André Castro Garcia

BANCA EXAMINADORA

_____
Prof. Esp. André Castro Garcia (Orientador)

_____
Prof. Me. Eduardo Machado Real

_____
Prof. Esp. José Gonçalves Dias Neto

NOVA ANDRADINA-MS
2018

DEDICATÓRIA

Dedico esse trabalho em primeiro lugar a Deus, e não menos importante a minha esposa Luana Crivelli e minha família que me apoiam desde o início, e em seguida os professores e os amigos que ao longo deste trajeto foram tendo sua participação garantindo assim que tudo isso fosse possível de ser realizado, sendo eles os mesmos que me incentivaram e tornaram possíveis as atividades e eventos relacionados a este projeto no qual fiquei muito satisfeito com seu resultado final.

AGRADECIMENTOS

Os mais sinceros agradecimentos e gratidão a Deus que esclarece meus ideais, permitindo cumprir essa conquista que é o meu projeto final.

Em primeiro lugar, gostaria de dizer que tem sido uma honra ser aluno desta Instituição de Ensino, à Universidade Estadual de Mato Grosso do Sul, e também gostaria de agradecer ao professor de meu último ano de supervisão do projeto o Sr. André Garcia pela orientação e apoio.

Por último, mas definitivamente muito importante, gostaria de agradecer a minha esposa e a minha família mãe, pai, e amigos pelo constante encorajamento e apoio que me ajudam a continuar e nunca desistir.

De todo coração, muito obrigado a todos vocês.

“Algumas pessoas acham que foco significa dizer sim para a coisa em que você vai se focar. Mas não é nada disso. Significa dizer não às centenas de outras boas ideias que existem. Você precisa selecionar cuidadosamente.”

Steve Jobs, em 2008.

SILVA, Silvano Crivelli. **FUTURO SOLAR: UM SERVIDOR AUTÔNOMO ALIMENTADO POR ENERGIA SOLAR.** Trabalho de Conclusão de Curso. Computação, Licenciatura. UEMS – Universidade Estadual de Mato Grosso do Sul – Unidade Universitária de Nova Andradina. Nova Andradina-MS. 2018.

Resumo: Este trabalho apresenta a implementação de um Sistema Autônomo alimentado com energia solar. O estudo destina-se satisfazer as necessidades de um conjunto de dispositivos de forma que sua fonte energética seja apenas a energia solar. Além disso, a proposta de um sistema Autônomo de controle residencial é mudar a ideia que temos de conforto em nossas casas, criando a possibilidade de controlar coisas remotamente. O sistema principal trata-se de um dispositivo Arduino que fará leituras de componentes interligados a ele, mostrando informações sobre energia renovável adquirida por painel fotovoltaico e por controle de reles que acionará dispositivos na casa. O visual do sistema não é o foco deste projeto, mas sim as informações apresentadas nele. O estado dos dispositivos controlados é sincronizado com a interface existente mostrando em tempo real a situação de cada um. O sistema tem seu projeto focado em ser autossuficiente em adquirir sua própria energia consumida e controlar dispositivos elétricos na casa de forma simples, e ainda poder alimentar lâmpadas com a mesma energia captada.

Palavras-chave: Arduino, Automação Residencial, Energia Solar, Sistema Autônomo, Sistema Autossuficiente, Renovável.

SILVA, Silvano Crivelli. **FUTURO SOLAR – SERVIDOR AUTÔNOMO ALIMENTADO POR ENERGIA SOLAR.** Trabalho de Conclusão de Curso. Computação, Licenciatura. UEMS – Universidade Estadual de Mato Grosso do Sul – Unidade Universitária de Nova Andradina. Nova Andradina-MS. 2018.

Abstract: This work presents the implementation of an Autonomous System powered by solar energy. The study is intended to meet the needs of a set of devices so that its source energy is only solar energy. In addition, the proposal of an autonomous system of residential control is to change the idea that we have comfort in our homes, creating the possibility of controlling things remotely. The main system is an Arduino device that will read component parts interconnected to it, showing information about renewable energy acquired by photovoltaic panel and control of relays that have triggered devices in the house. The system look is not the focus of this project, but rather the information presented in it. The state of the controlled devices is synchronized with the existing interface showing in real time the situation of each one. The system has its project focused on being self-sufficient in acquiring its own consumed energy and controlling electrical devices in the house in a simple way, and still being able to feed light bulbs with the same energy captured.

Keywords: Arduino, Residential Automation, Solar Energy, Autonomous System, Self-sufficient, Renewable.

LISTA DE FIGURAS

Figura 1 Painel Fotovoltaico....16
Figura 2 Arduino UNO....23
Figura 3 Adaptador USB Serial....24
Figura 4 Componente RS232 DB9....24
Figura 5 Placa de Relês....25
Figura 6 LCD 2004 20x4 Display....25
Figura 7 Medidor de Amperes....26
Figura 8 Thin client....27
Figura 9 Resistores....28
Figura 10 Circuito Voltímetro....29
Figura 11 Bateria....30
Figura 12 Regulador de Voltagem....31
Figura 13 Jumper e Protoboard....31
Figura 14 Estrutura resumida....33
Figura 15 Código checar tensão....34
Figura 16 Sources.list....36
Figura 17 Comandos update e install....36
Figura 18 Código PHP....37
Figura 19 Inserção do PHP....38
Figura 20 Condição if....39
Figura 21 Leitura Serial....40
Figura 22 Exemplo de código strpos( ) substr( )....40
Figura 23 Percentual da bateria....41
Figura 24 Condições PHP Bateria....41
Figura 25 Corrente de carga....42
Figura 26 Página Web do Sistema....43
Figura 27 Código formulário....44

SUMÁRIO

1 INTRODUÇÃO 12
1.1 MOTIVAÇÃO 13
1.2 OBJETIVOS 13
1.2.1 Objetivo Geral 13
1.2.2 Objetivos Específicos 14
1.3 METODOLOGIA 15
1.3.1 Fluxo do projeto 15
1.3.2 Energia Solar: Painel Fotovoltaico 15
2 FUNDAMENTAÇÃO TEÓRICA 18
2.1 INTERNET DAS COISAS 18
2.2 SISTEMAS EMBARCADOS 19
2.3 ARDUINO 19
3 SISTEMA AUTÔNOMO DESENVOLVIDO 21
3.1 VIABILIDADE 21
3.2 ESCOLHA DO HARDWARE 21
3.3 PROBLEMA DE PESQUISA 22
3.4 TECNOLOGIAS E COMPONENTES 23
3.4.1 Interface de programação Arduino IDE 23
3.4.2 Arduino UNO 23
3.4.3 Adaptador USB Serial 24
3.4.4 Componente RS232 24
3.4.5 Placa de Relês 25
3.4.6 Lcd 2004 20x4 Display 25
3.4.7 Medidor de Amperes 26
3.4.8 Thin client – Mini Computador 27
3.4.9 Medidor de Tensão 28
3.4.10 Bateria e Armazenamento 29
3.4.11 Regulador de Tensão 30
3.4.12 Jumper e Protoboard 31
3.5 COMUNICAÇÃO 32
3.6 IMPLEMENTAÇÃO DE HARDWARE 32
3.7 SERVIDOR DE APLICATIVOS WEB PHP 34
3.7.1 Por que usar um servidor Web 34

3.7.2 Criação do Servidor Web PHP 35
3.7.3 Funcionamento básico do PHP no Servidor Apache. 37
3.7.4 Programando instruções *if* em PHP para representação de imagens. 40
3.7.5 Programando botões para ativar dispositivos com reles. 43
4 CONCLUSÃO 45
REFERÊNCIAS 46
ANEXO I 48
ANEXO II 54

1 INTRODUÇÃO

A energia solar é claramente a forma mais limpa de energia renovável, e pode ser usada de diversas formas. Esta proposta de sistema de automação residencial poderá trazer consigo este diferencial, pois ele poderá ter a capacidade de renovar sua energia consumida através da captação e armazenamento desta abundante matéria prima em nosso país, que é o Sol. Sua luz será aproveitada neste projeto, a captação de energia solar por painel fotovoltaico poderá trazer uma imensa vantagem neste sistema. Sobre a automação de funções comuns em residências, como ligar lâmpadas e dispositivos, serão mais fáceis pois existem vários exemplos na rede mundial de computadores à internet. Tornar objetivos de controle mais práticos onde remotamente poderá apenas selecionar um comando, traz vantagens para os usuários, ainda mais para os usuários deficientes o que facilitará o controle de seus eletrodomésticos.

Outra diferença com sistemas de automação residencial e que este pode funcionar mesmo sem energia elétrica de uma concessionária local, ligando lâmpadas e dispositivos alimentados pela energia captada e/ou armazenada da energia solar. Claro que os aparelhos conectados devem ser compatíveis com a energia captada e estarão ligados a um controlador central que combinados com outros dispositivos mostraram informações remotamente e farão controle dos mesmos.

Hoje em dia todos têm e fazem uso de um Smartphone o qual facilita o acesso à internet sendo esse o meio de controle do sistema desenvolvido em que mostrará as informações e possibilitará o desenvolvimento de suas funções. Um estudo postado pela Folha UOL[1] realizado abril de 2016 mostra que o Brasil terá aproximadamente 168 milhões de Smartphones com projeção que até 2018 este número chegue a 236 milhões de aparelhos e destes, 85% deles teriam sistema Android. Pensando nesta linha de raciocínio será fácil tem em mãos um dispositivo portável que faz uso de um navegador, possibilitando assim o acesso à internet.

[1] http://www1.folha.uol.com.br/mercado/2016/04/1761310-numero-de-smartphones-em-uso-no-brasil-chega-a-168-milhoes-diz-estudo.shtml

1.1 MOTIVAÇÃO

Nosso país possui grandes recursos naturais e se fossem mais analisados em centros de pesquisa poderiam gerir o desenvolvimento de novas tecnologias incríveis, o site Americadosol[2] possui muita informação sobre o território brasileiro. Segundo o site o Brasil possui um grande potencial para gerar eletricidade a partir do sol. Só para se ter uma ideia, a radiação solar na região mais ensolarada da Alemanha, por exemplo, que é um dos líderes no uso da energia fotovoltaica (FV), é 40% menor do que na região menos ensolarada da Brasil. Segundo o Atlas Brasileiro de Energia Solar, diariamente incide entre 4.444 Wh/m^2 a 5.483 Wh/m^2 no país. Apesar dessas condições favoráveis, o uso de energia solar para geração elétrica ainda é pouco considerado como uma opção para alimentar indústrias, casas e edifícios. Como o país já possui uma das matrizes energéticas mais limpas do mundo, a melhor integração da energia solar FV vem sendo utilizada como fonte complementar, aproximando a geração do consumo e reduzindo assim perdas com transmissão.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Criar um sistema autossuficiente que não use a energia elétrica convencional, este sistema terá seu funcionamento sob captação de energia solar e armazenará a energia excedente para utiliza-la no período noturno, deverá captar e processar informações como quantidade de energia gerada WG (Watts gerado) e quantidade de energia consumida WC (Watts consumido). Deverá ter também outras funções como ativar e desativar dispositivos diversos instalados em uma casa (lâmpadas, ventiladores etc.) e estas informações e controles serão disponibilizados via internet com utilização de softwares Open Source. Outro foco abordado neste projeto é estimular novas ideias de programação com relação em sistemas autônomos, criando um novo campo de possibilidades, e mostrando que é possível interagir com o meio criando e utilizando energia limpa.

[2] http://americadosol.org/potencial-solar-no-brasil#toggle-id-1

1.2.2 Objetivos Específicos

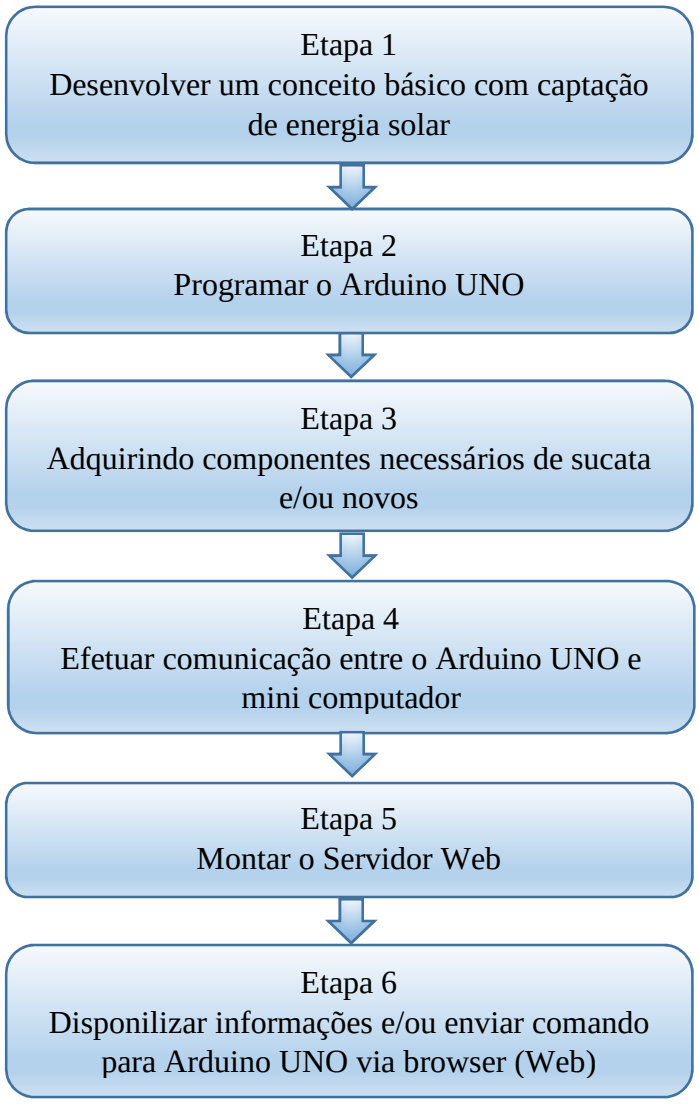
Automatizar dispositivos que podem controlar iluminação e eletrodomésticos utilizando uma interface Web para observar e controlar o sistema. Para cumprir o objetivo geral, deverá cumprir requisitos especificos que envolvem programação de software e implementação de hardware. Estas podem ser divididas da seguinte forma:

- Criar um sistema autossuficiente que não necessite usar energia convencional, a energia será gerada com placa de Energia Solar Fotovoltaica sendo essa a única fonte de alimentação elétrica.
- Captar energia limpa renovável com painéis fotovoltaicos tornando o sistema autossustentável.
- Gerenciar a captação de energia coletando dados através dos sensores conectados à placa Arduino.
- Estabelecer uma comunicação entre a placa Arduino e o mini computador através de componentes compatíveis entre os dois dispositivos.
- Criar um sistema de automação residencial simples, usando a placa Arduino que será o meio de controle para ativar dispositivos e/ou eletrodomésticos.
- Encontrar as TICs adequadas que funcione eficientemente com a placa Arduino.
- Programar a placa Arduino de uma forma que permita interagir com o sistema Web recebendo dados pela porta Serial.

1.3 METODOLOGIA

O conteúdo deste capítulo tem a finalidade de esclarecer os metodos a serem alcançados para o desenvolvimento deste projeto.

1.3.1 Fluxo do projeto



1.3.2 Energia Solar: Painel Fotovoltaico

De acordo com o fluxograma etapa 1, as células solares, também chamadas de células fotovoltaicas (FV) pelos cientistas, convertem a luz solar diretamente em eletricidade.

Segundo (Messenger e Ventre, 2010, p.49) as células FV recebe o nome do processo de conversão de luz (fótons) para eletricidade (tensão). Em uma célula fotovoltaica, a luz excita elétrons para passar de uma camada para outra através de materiais de silício semicondutores, isso produz uma corrente elétrica.

A fotocorrente criada em uma célula fotovoltaica é dependente da intensidade da luz incidente na célula. A fotocorrente também é altamente dependente do comprimento de onda da luz incidente. As células geralmente têm superfícies texturizadas e também podem ser revestidas com um revestimento antirreflexo para minimizar a reflexão da luz solar longe das células.

Células solares chamadas fotovoltaicas são feitas de finas fatias de silício cristalino, gálio arseneto, ou outros materiais semicondutores, convertem a radiação solar diretamente em eletricidade. As células solares mais simples fornecem pequenas quantidades de energia para relógios e calculadoras. Sistemas mais complexos podem fornecer eletricidade para casas e redes elétricas. Normalmente, porém, as células solares fornecem baixa potência para dispositivos remotos e sem supervisão, como bóias, satélites meteorológicos e de comunicação e equipamentos a bordo de espaçonaves.

Uma certa quantidade de células solares conectadas umas às outras e montadas em uma estrutura ou suporte podemos chamar de módulo fotovoltaico (Figura 1). Os módulos são projetados para fornecer eletricidade a uma determinada tensão, para isso basta ligar as células eletricamente uma as outras, criando um sistema comum de 12 volts de corrente continua. A corrente produzida é dependente da quantidade de luz que atinge o módulo.

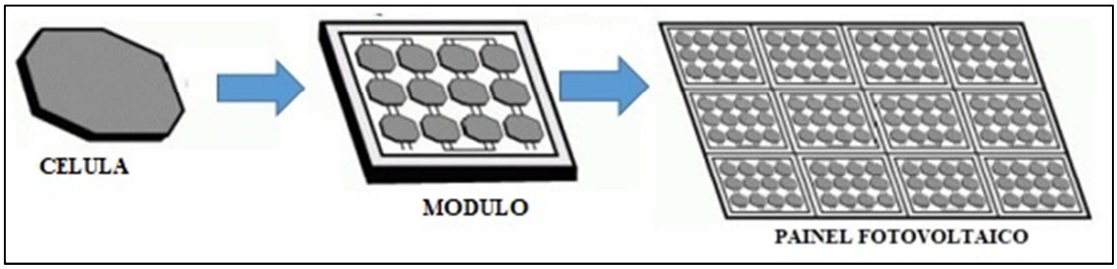


Figura 1 Painel Fotovoltaico
Fonte: Do Autor

Vários módulos podem ser conectados para formar Painel Fotovoltaico. Em geral, quanto maior a área de um módulo mais celulas existirão ali, e mais eletricidade será produzida. Módulos e matrizes fotovoltaicos produzem eletricidade de corrente contínua

(DC). Estes módulos podem ser conectados em arranjos elétricos ligando-os em série ou paralelo para produzir qualquer combinação de tensão e corrente necessária.

Para este projeto, o conjunto de módulos presente no painel fotovoltaico adquirido é capaz de gerar uma potência de até 55W, e a tensão máxima de saída pode chegar até 17,8 volts com uma corrente máxima de 3,08A. Com boas condições de tempo, se obtivermos 5 horas de sol o painel pode gerar 275W e/ou 15,4A de carga para ser armazenada em uma bateria. Se o sistema autônomo mantiver seu consumo abaixo de 1A isso nos daria condições de mantê-lo funcionando por mais de 15 horas sem a necessidade de recarga. Se pensarmos na seguinte situação, que após as 17 horas o sistema não estará mais armazenando e sim consumindo, e sabendo que somente no dia seguinte a partir das 6 horas da manhã o sol poderá estar sendo captado pelo painel solar, podemos calcular 13 horas de consumo e assim deduzir que a carga adquirida de 15,4A será suficiente para manter o sistema funcionando durante o período noturno. Tendo chegado a este simples conceito podemos então dar seguimento as demais fases do projeto.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 INTERNET DAS COISAS

A facilidade de conexão com a internet hoje, facilita o desenvolvimento de muitas tecnologias e vem tornando possível cada vez mais desenvolver tecnologias que poderão, e muito provavelmente já estão tornando possível realizar tarefas à distância, utilizando a internet e enviando informações à sua residência para que algum dispositivo realize algo para você. Uma melhor definição para estas tecnologias são:

> Desde há bastantes anos que se imaginam e implementam soluções baseadas no conceito de ligação de dispositivos à Internet, muito para além dos tradicionais computadores, que foram os primeiros meios de acesso. O que há alguns anos era um conceito é hoje uma realidade em concretização, com muitos meios, pessoas e ideias envolvidos: a IoT (Internet of Things – Internet das Coisas) chegou e vai tomar conta do nosso dia a dia. COELHO (2017, pág. 1)

Entretanto, Stanlling também conceitua que a Internet das coisas pode ser definida:

> É um termo que se refere a interconexão expansiva dos dispositivos inteligentes, indo de aplicações a minúsculos sensores. Um tema dominante é a incorporação de transceptores moveis de baixo alcance em uma grande variedade de dispositivos e itens do dia a dia, possibilitando novas formas de comunicação entre pessoas e as coisas entre si. A internet agora apoia a interconexão de bilhões de objetos industriais e pessoais geralmente por meio de sistemas em nuvem. STANLLING (2017, p.25)

Variadas são as definições para a IoT, mas essa área tem colocado muitos dispositivos na Internet, e esses dispositivos são basicamente objetos interagindo com outros objetos. Essa interligação entre dispositivos podem se comunicar de forma autônoma e efetuar tarefas sem a necessidade de interação humana, mas claro que estas ações seriam definidas em suas programações que por fim pressupõe-se criado por humanos. Um dos principais hardwares utilizado por muitos desenvolvedores esta presente na placa Arduino, ela facilita a programação e compilação com o hardware, e tem um software Arduino IDE[3] já preparado para esta função, e de acordo com o fluxograma na etapa 2, a programação se inicará. O Arduino tem também uma gama muito grande de dispositivos e componentes compatíveis agilizando o desenvolvimento e possibilitando a construção dos mais diversos objetos inteligentes conectados.

[3] Arduino IDE é a plataforma de programação de código aberto para o Arduino, que faz interface entre o usuário e o hardware, torna fácil escrever o código e envia-lo para a placa.

2.2 SISTEMAS EMBARCADOS

Segundo (Stalling, 2017, p.24) os sistemas embarcados[4] vêm da necessidade de satisfazer a complexidade em desenvolver dispositivos eletrônicos mais inteligentes substituindo os eletrônicos sem habilidade computacional. À uma necessidade cada vez maior de viver em um mundo conectado onde o mundo virtual começa a fazer parte do mundo real, os Sistemas Embarcados começam a tornar possível essa interação.

Outra facilidade é que Sistemas embarcados estão se tornando mais baratos, consumindo menos energia, estão cada vez menor e seu poder de processamento cada vez maior. Pensando nesta linha de evolução de forma constante, com o passar do tempo estaremos rodeados por milhões de dispositivos conectados que trarão uma nova forma de interagir com o ambiente real a nossa volta.

2.3 ARDUINO

O Arduino é uma plataforma eletrônica de código aberto e, segundo em seu site oficial www.arduino.cc, é baseada em hardware e software fáceis de usar. A placa Arduino é capaz de ler entradas, tais como a luz de um sensor, um dedo em um botão ou uma mensagem no Twitter, e transformá-las em uma saída ativando um motor, ligando um LED ou publicando algo online. É possível dizer à sua placa o que fazer, enviando um conjunto de instruções para o microcontrolador[5].

O Arduino é fácil de usar para iniciantes, mas flexível o suficiente para que usuários avançados aproveitem também. Para os professores, ele é convenientemente baseado no ambiente de programação, para que os alunos que aprendem a programar nesse ambiente estejam familiarizados com o funcionamento da Arduino IDE disponivel. O Arduino também simplifica o processo de trabalho com microcontroladores, mas oferece algumas vantagens para professores, alunos e amadores interessados em outros sistemas.

O software Arduino IDE é publicado como ferramentas Open Source, disponível para extensão por programadores experientes. A linguagem pode ser expandida através de

[4] Sistemas embarcados são dispositivos que tem a sua programação criada para um único propósito de sua aplicação, essa programação é salva em uma memória especifica dentro do microcontrolador.
[5] Um microcontrolador é o um sistema micro processado com várias funcionalidades (periféricos) disponíveis em um único chip. Basicamente, um microcontrolador é um microprocessador com memórias de programa, de dados e RAM, temporizadores e circuitos de *clock* embutidos.

bibliotecas C ++. O projeto AVR[6] tem uma CPU de 8 bits, suporte digital básico para E/S, serial e entradas analógicas.

Uma definição também importante sobre microcontroladores veem do conceito detalhado que define o microcontrolador:

> [...] os computadores embutidos, as vezes denominados microcontroladores, gerenciam os dispositivos e manipulam a interface de usuário. São encontrados em grande variedade de aparelhos diferentes [...] o sistema Arduino é um projeto de hardware de fonte aberta, o que significa que todos os seus detalhes são publicados e gratuitos, de modo que qualquer um pode montar (e até mesmo vender) um sistema Arduino. Ele é baseado no microprocessador RISC de 8 bits Atmel AVR, e na maioria dos projetos de placa também inclui suporte básico para E/S. TANENBAUM, (2013, p.26 e 27)

Tendo estas informações como contexto, a placa Arduino UNO torna-se à escolha de hardware mais óbvia e eficaz para este projeto com seu conteúdo vasto de exemplos palpáveis.

[6] AVR são microcontroladores de arquitetura RISC – *Reduced Instruction Set Computer* (Computador com Set de Instrução Reduzido) com memoria interna, baixo custo e consumo reduzido. TANENBAUM,(2013)

3 SISTEMA AUTÔNOMO DESENVOLVIDO

3.1 VIABILIDADE

Para garantir o baixo custo deste sistema autonomo e ser possível sua criação foi necessário pesquisar recursos de hardware capazes de realizar as tarefas pretendidas e que tivessem um baixo custo. Durante a pesquisa de preços, constatou-se que os produtos adquiridos em lojas nacionais ficariam até três vezes mais caros que os mesmos produtos sendo importados, mesmo considerando os custos de transporte. Já que a aquisição destes produtos tecnológicos é de suma importância para a realização do projeto, foram importados alguns componentes como, por exemplo, a placa Arduino, os sensores para as medições de carga e os reles para ativar e desativar recursos externos.

3.2 ESCOLHA DO HARDWARE

Neste projeto foi escolhida a placa Arduino UNO com um conversor USB RS232[7] com pino DB9 para fazer a comunicação com o mini computador. O custo desta placa Arduino é extremamente baixo e fácil de encontrar, e este modelo é compatível com uma gama enorme de componentes já existentes no mercado para desenvolvimento de tecnologias TICs[8]. Na placa Arduino UNO as informações são recebidas via conversor USB RS232 e são processadas pelo micro controlador que respondera de acordo com a programação. O display utilizado neste projeto trata-se do modelo LCD 2004 20x4 onde algumas informações de controle serão mostradas diretamente nesta interface de hardware. Outro motivo para escolher o Arduino esta muito bem apresentada por Lima e Vilhaça, (2012), e cito uma parte muito importante para o desenvolvimento deste projeto, sendo:

> A IDE apresenta tal grau de abstração, que o usuário não precisa saber o que é um microcontrolador, nem como se utilizam seus registradores de trabalho. Aliado a essa característica, o hardware do Arduino apresenta um suporte crescente na comunidade técnica, com a disponibilidade de bibliotecas de funções para as mais complexas atividades. Tudo associado a um número cada vez maior de módulos prontos para serem conectados ao Arduino. LIMA & VILHAÇA, (2012, p.499).

[7] RS232 é um componente de comunicação serial presente nos computadores pessoais.
[8] TIC Tecnologia da Informação e comunicação pode ser definida como um conjunto de recursos tecnológicos, que proporcionam um novo modo de se comunicar.

Outro hardware também utilizado neste projeto trata-se de um mini computador modelo Thin Client [9] adquirido em sucata, este modelo de computador é muito limitado mais seus recursos são mais que suficientes para a demonstração do servidor neste projeto.

3.3 PROBLEMA DE PESQUISA

O problema levantado foi á criação de um sistema autônomo capaz de funcionar sem energia convencional e automatizar funções de uma residência com propósito de melhorar essa interação. A intenção é criar um sistema que ofereça um modo de vida futurista no qual um indivíduo controla toda a sua casa usando um smartphone, podendo ligar lâmpadas e dispositivos eletrônicos e até abrir portas.

Ele também oferece um uso de energia renovável, sendo autossuficiente em se reabastecer. No entanto sabe-se que para obter ou adquirir componentes para montagem e criação de tal sistema é necessário um custo a ser analisado o qual seria a principal razão de uma dificuldade para sua implantação, adicionando a isso a complexidade de detalhes para instalá-lo e configurá-lo. Tendo esta visão, é necessário torná-lo rentável para facilitar a aquisição dos componentes e dispositivos, deve-se também ser de fácil uso, concedendo às pessoas a possiblidade de entender o sistema rapidamente.

O aproveitamento de componentes e até mesmo de um mini computador de sucata podem trazer à este projeto o conceito de torná-lo mais rentável diminuído assim os custos de implantação do mesmo. Em outras palavras, não se deve focar apenas no desenvolvimento do sistema de automação em si, mas também se preocupar com a necessidade de reduzir os custos para aplicá-lo em casas assim a automação residencial visa oferecer facilidades para todos os tipos de indivíduos tendo entre eles o deficiente físico que seria um dos principais beneficiários, realizando com apenas um clique em seu smartphone tarefas que poderiam ter certas dificuldades em realiza-lás.

[9] O Thin Client é um equipamento que funciona como um mini PC, mas não possui, em sua estrutura interna, HD, processador e memória (não como os convencionais). Apesar de sua estrutura simples, com ele é possível obter uma rede de baixo custo e de fácil manutenção, dentre outros benefícios. (BARNABÉ, ROCHA, VIEIRA & SOUZA, 2010).

3.4 TECNOLOGIAS E COMPONENTES

3.4.1 Interface de programação Arduino IDE

Por razões óbvias, já que o software é livre, e disponibilizado pela mesma empresa da placa do microcontrolador Arduino, o software de programação escolhido será o Arduino IDE. A escolha desta interface é por conta de já estar preparada para este tipo de programação e também compilar e enviar o código para o circuito do microcontrolador. Outro fator é que o site oficial do Arduino disponibiliza bibliotecas e modelos reais de aplicações usadas pelo código. A linguagem de programação utilizada é um conjunto de funções C/C+ + que podem ser chamadas a partir do seu código com pequenas alterações e, em seguida é passado diretamente em um compilador C/C+ +.

3.4.2 Arduino UNO

O primeiro item desta etapa 3 do fluxograma, é a placa Arduino UNO (Figura 2), escolhida devido a facilidade de uso. Sua plataforma de código aberto em hardware e software traz uma infinidade de exemplos e componentes de fácil aquisição, e isso tem sido a razão de muitos projetos serem desenvolvidos com este microcontrolador. Seu modelo foi desenvolvido para ser de fácil aprendizagem, sendo muito utilizado em sua maioria por estudantes de eletrônica e programação. A variedade enorme de componente garante que o usuário consiga muito facilmente adapta-lo a qualquer necessidade especifica. Seu software por se tratar de condigo aberto, garante um crescimento enorme de contribuições de usuários no mundo todo.

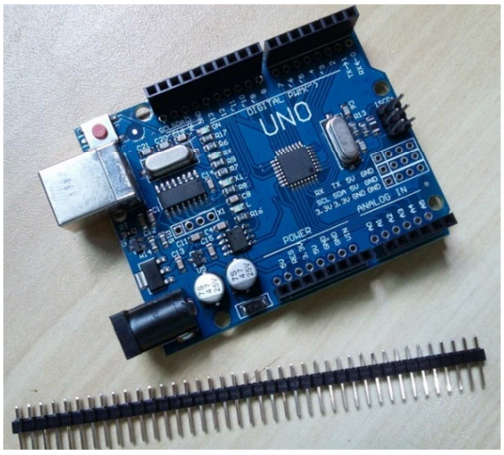


Figura 2 Arduino UNO
Fonte: Do Autor

3.4.3 Adaptador USB Serial

O adaptador USB Serial (figura 3) foi escolhido pra facilitar a compatibilidade com o mini computador Thin Cliente, pois ele não possui porta Serial mas por compensação possui quatro (4) portas USB e uma delas será utilizada neste dispositivo. O adaptador Serial será conectado na saída RS232 do Arduino e será possível enviar e receber dados.

Figura 3 Adaptador USB Serial
Fonte: Do Autor

3.4.4 Componente RS232

Esta forma de comunicação já vem de muito tempo atrás, era muito utilizada entre computadores e periféricos. A porta serial do computador contem pinos que enviam e recebem informações, que são de fácil leitura e isso facilitará a programação deste sistema. A figura 4 mostra o componente RS232. Um fato pelo qual foi necessário a utilização deste componete é que porta RS232 virtual, padrão da placa Arduino, estava reinicializando toda vez que uma solicitação de litura era efetuada, e com a utilização deste componente este problema foi eliminado.

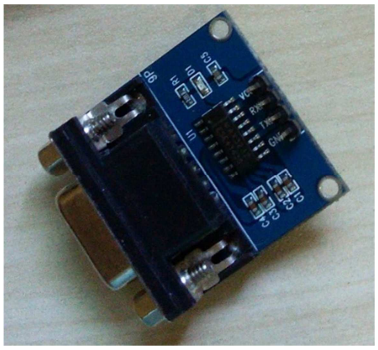


Figura 4 Componente RS232 DB9
Fonte: Do Autor

3.4.5 Placa de Relês

Esta placa (figura 5) será utilizada para controle de outros dispositivos, com ela e possível trabalhar como chaves de liga e desliga. A placa é compatível com o microcontrolador Arduino e para isso basta enviar valores através de sua programação para mudar o estado das chaves (relês), isso fará com que ligue ou desligue uma lâmpada ou um dispositivo eletrônico qualquer. Esta placa possui 4 (quatro) relês com chaveamento independente.

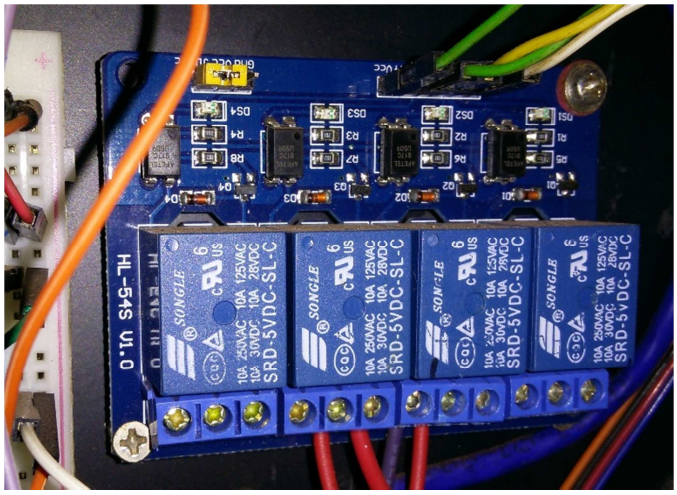


Figura 5 Placa de Relês
Fonte: Do Autor

3.4.6 Lcd 2004 20x4 Display

O uso do display LCD 20x4 (figura 6) facilita a programação, pois possibilita saber o estado do microcontrolador, mostrando os estados de controle dos dispositivos e chaveamento dos relês, assim como informações do medidor de carga, corrente da bateria e etc. O display possui um espaço limitado e apenas informações essências serão apresentadas nele. O modelo escolhido também é compatível com o Arduino.

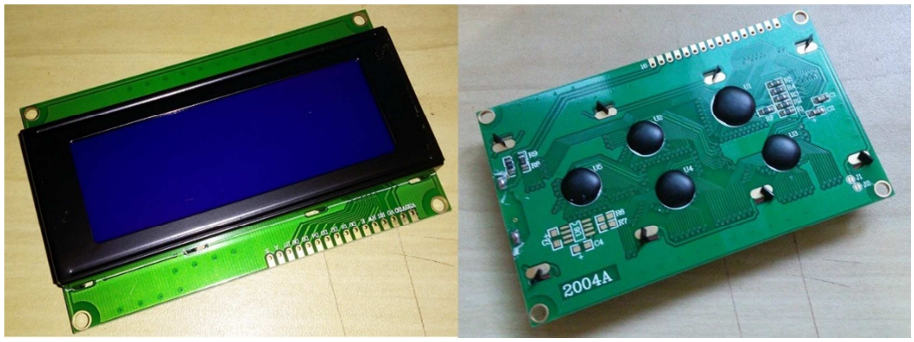


Figura 6 LCD 2004 20x4 Display
Fonte: Do Autor

3.4.7 Medidor de Amperes

O sensor para medir a carga da corrente continua DC possui um mini circuito eletrônico como é o caso do ACS712 5Ah (figura 7), modelo este escolhido para o projeto. Para correntes alternadas bastaria apenas uma bobina com centenas de voltas de fio de cobre e assim seria possível fazer a leitura.

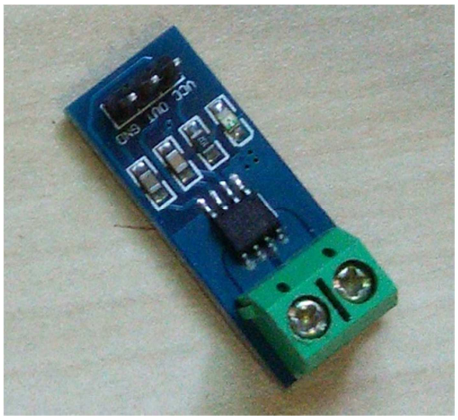


Figura 7 Medidor de Amperes
Fonte: Do Autor

O componente ACS712 faz leituras de -5A até +5A, o que também dificulta um pouco sua leitura precisa, pois o projeto está trabalhando com consumo e carga máxima perto da casa dos 2A (dois amperes). O por que não ser tão simples efetuar a leitura é devido ao componente responder com valores digitais que variam de 0 a 1023 com uma razão de saída, que de acordo com seu fabricante, deveria ser de 185mV/A (mV/A, microvolts por Amperes),

Explicando um pouco mais sobre os valores de resposta do componente significa que este ira mapear tensões ente 0 e 5 volts para valores inteiros entre 0 e 1023, que neste caso trabalham com -512 para leitura negativa e +512 para positiva. Isso permite leituras de 0,0049 volts (4,9 mV) por unidade. A fórmula (mV) pode ser descrita na Equação:

$$mV = \frac{volts}{unidades}$$

Foi realizado leituras do componente lado a lado com um multímetro digital para verificar a precisão da leitura. Notou-se durante os testes que o ACS712 oscila sua medição de 0,2A para mais ou para menos.

3.4.8 Thin client – Mini Computador

Este modelo de computador foi projetado para realizar conexões em servidores de aplicativos remotos, onde ele mostra uma área de trabalho que na realidade está sendo processada em um computador com maior poder de processamento (servidor). O *Thin client* (figura 8) é o responsável pelo acesso remoto permitindo que o usuário utilize deste servidor para realizar suas tarefas em um ambiente de trabalho controlado. Para que tudo isso seja possível, o *Thin client* precisa ter uma interface de usuário e para isso ele precisa de um micro sistema que necessitará de processamento, memória e outros componentes que claramente torna sua arquitetura muito parecida com a de um computador, mas está claro que seus recurso são limitados e de longe ele não tem o mesmo potencial dos computadores atuais. Este mini computador tem as seguintes especificações; possui processador de 840Mhz e memória 512M, contém uma porta VGA (vídeo), quatro portas USB (Universal Serial Bus) e uma porta ethernet, sua alimentação é efetuada por corrente continua em 12volts com carga de 1Ah, o que facilitou muito a compatibilidade neste projeto por estarmos trabalhando com baterias recarregáveis de 12volts.

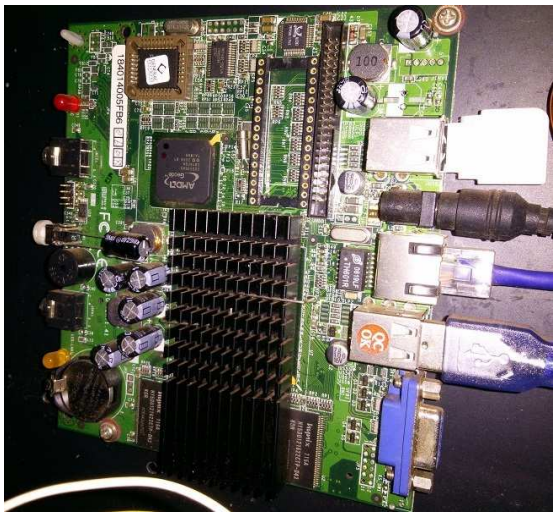


Figura 8 *Thin client*
Fonte: Do Autor

Obviamente por se tratar de uma máquina com recursos limitados ele ganha em alguns aspectos, como baixo consumo de energia, baixo custo de hardware e compatibilidade com sistemas comuns como o Linux. Isto neste projeto é muito importante, pois, quanto mais energia for economizada, mais estável e duradouro será o funcionamento do sistema. O sistema operacional Linux instalado será uma versão antiga do Ubuntu Server 10.04, pois ele

consome menos recursos de *hardware*, tornando melhor o desempenho das aplicações que desempenham o papél do servidor Web, que permitirá mostrar informações deste sistema autônomo.

3.4.9 Medidor de Tensão

O simples voltímetro foi construído com apenas duas resistores sendo eles R1 e R2 (figura 9). Esse pequeno circuito é capaz de efetuar leituras de até 55 volts sendo mais que suficiente para medir a corrente utilizada neste sistema que opera com 12 volts.

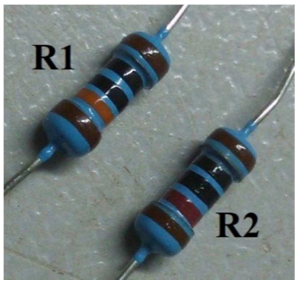


Figura 9 Resistores
Fonte: Do Autor

O circuito funciona com cálculos efetuados com os resistores, sendo R1 100K e o resistor R2 10k. Sabendo que o fator de medida da porta analógica do Arduino é de 1024 e a corrente máxima é de 5 volts, encontramos o valor da unidade que seria 5 / 1024 = 0,0049. Desta forma o valor de entrada no pino em que será realizada a leitura da voltagem deve ser multiplicado por 0,0049 (valor_input * 0,0049). Tendo todos esses valores anteriores sendo eles o R1, R2 e valor de leitura do pino A0 (pino analógico zero), pode-se extrair a tensão de acordo com a seguinte Equação:

$$Vin = Vout\ x \frac{(R1 + R2)}{R2}$$

Para finalizarmos o voltímetro usou-se o esquema circuito como mostra a figura 10, este por sua vez é muito simples onde uma das extremidades do R1 é feita a ligação de entrada da corrente, a outra extremidade fica conectada à extremidade do R2 e nesse ponto é feito a conexão com o pino analógico do Arduino para realizar a leitura e efetuar o cálculo que demonstrará a tensão atual que o circuito estará recebendo na entrada ^{V}in. Para conluir a ligação dos resistores, a outra extremidade do resistor R2 deve ser ligada ao negativo do

circuito. Este circuito reduz as tensões de entrada para correntes compatíveis com o Arduino, desta forma a corrente de saída Vout não ultrapassará valores capazes de danificar o microcontrolador, e será utilizado no cálculo da tensão. Os desenhos de circuitos criados a partir daqui foram feitos com ajuda do site CIRCUITLAB[10].

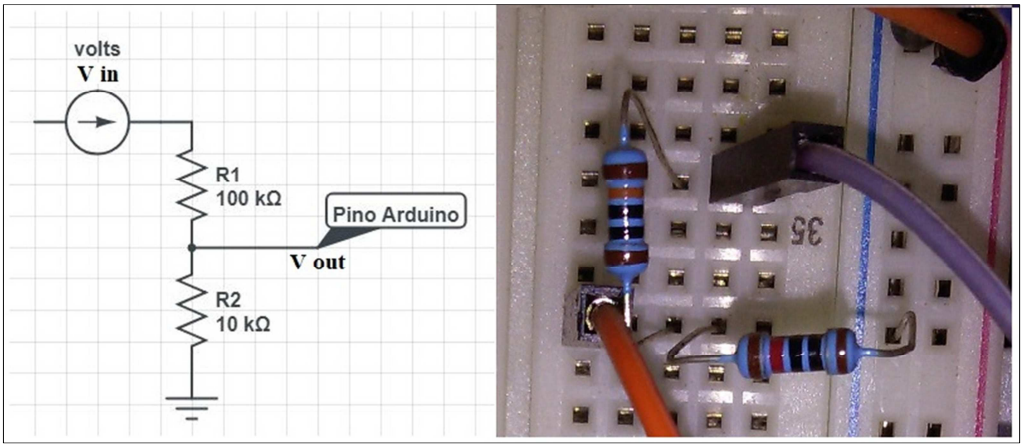


Figura 10 Circuito Voltímetro
Fonte: Do Autor

A medição da tensão será utilizada neste sistema para realizar outros cálculos. O valor da tensão multiplicado pelo ampere, o qual está sendo recebido do sensor ACS712, nos dará uma ideia da potencia em Watts na geração de energia. A fórmula é apresenta neste exemplo de Equação:

$$Watts = volts * Ah$$

Embora a capacidade de uma bateria seja normalmente definida como a quantidade de amperes-hora (Ah) que pode ser retirada da mesma quando esta apresenta carga plena, a capacidade de uma bateria também pode ser expressa em termos de energia (watts-hora). (PINHO e GALDINO, 2014).

3.4.10 Bateria e Armazenamento

Devido a algumas necessidades de desenvolvimento deste sistema autônomo, o armazenamento de energia faz se necessário, e como a aquisição desta energia renovável depende exclusivamente de luz solar, será necessário manter o sistema funcionando quando não estiver disponível. Praticamente todas as noites e dias de chuva com pouca ou nenhuma luz solar tornarão a captação de enernegia impossivel e certamente o sistema poderá falhar. A

[10] https://www.circuitlab.com/editor

bateria (figura 11) torna-se então necessária, pois ela fara o papel de alimentar o sistema durante períodos onde a energia solar não estará disponível.

Figura 11 Bateria
Fonte: Do Autor

A bateria que está sendo usada tem tensão de 12 volts e 18Ah, foi adquirida em sucata de um nobreak descartado por defeito elétrico, e foi necessário uma manutenção na bateria para melhorar seu desempenho. Para definir melhor o que é uma bateria podemos dizer que:

> [...] refere-se a um grupo de células eletroquímicas conectadas eletricamente em serie e/ou paralelo, para produzir tensão e/ou corrente mais elevados do que a que pode ser obtida por uma única célula. Uma bateria pode também ser construída por uma única célula, caso esta constitua em um sistema de armazenamento eletroquímico completo. Assim uma bateria e um dispositivo eletroquímico que converte energia química em energia elétrica e vice-versa. [...] a capacidade de uma bateria seja normalmente definida como a quantidades de amperes-hora (Ah) que pode ser retirada da mesma quando esta apresenta carga plena, a capacidade de uma bateria também pode ser expressa em termos de energia (watts-hora). PINHO & GALDINO (2014, p166)

Portanto a capacidade desta bateria e suficiente para o funcionamento deste sistema já que todo os dispositivos sem o acionamento das luzes, estão consumindo menos de 1Ah (um ampere hora) e isso nos daria uma autonomia de funcionamento de aproximadamente 18 horas sem a recarga da bateria.

3.4.11 Regulador de Tensão

Devido à conexão de alimentação auxiliar do Arduino ser de 9 volts faz-se necessário o uso de um regulador de tensão. O regulador escolhido foi o Transistor LM7809 (figura 12)

que trabalha com tensões de entrada de 11.5 a 26 volts e suporta carga de 1A e sua saída de tensão será de 9 volts. Um pequeno dissipador de calor ajudara o manter arrefecido para não sofrer dano.

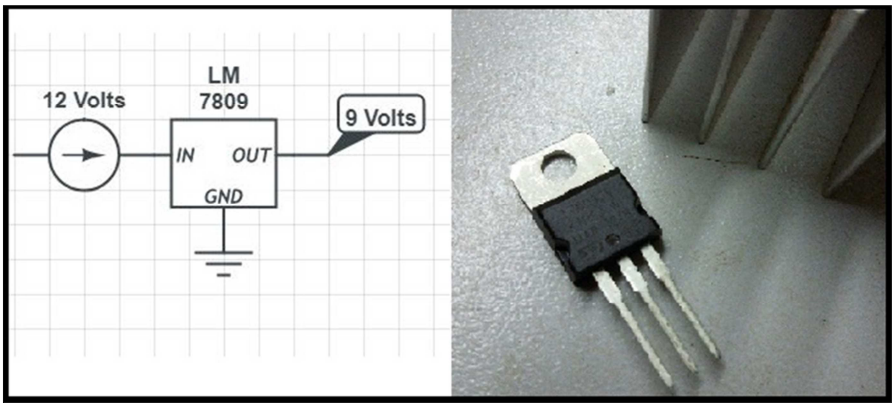


Figura 12 Regulador de Voltagem
Fonte: Do Autor

A saída de corrente será ligada apenas na alimentação auxiliar do Arduino, e as demais correntes para alimentar os componentes são as saídas de alimentação do próprio microcontrolador. O datasheet deste transistor pode ser visto no site SquereSpace.com.

3.4.12 Jumper e Protoboard

Jumpers são simples ligações, efetuam contatos elétricos (curto) entre pontos distintos, e esses são realizados por fios comuns. Esses fios faram ligações entre peças e componentes eletrônicos usados neste projeto. A figura 13 dá uma ideia destas ligações.

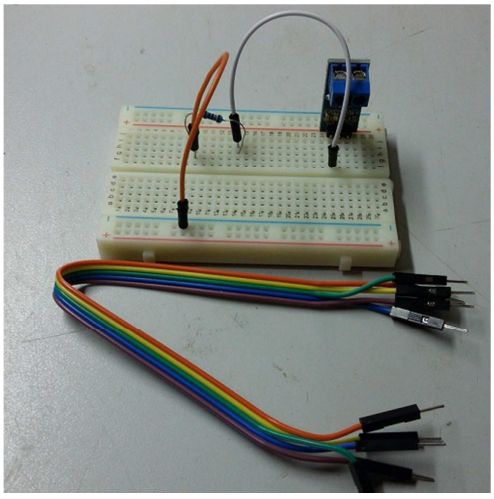


Figura 13 Jumper e Protoboard
Fonte: Do Autor

3.5 COMUNICAÇÃO

A intercomunicação entre o computador e o microcontrolador faz-se necessária neste projeto, e seguindo o fluxograma na etapa 4, trata a interconexão entre a placa Arduino e o mini computador. Segundo (Stallings, 2017, p.194) as operações de entrada e saída de dados E/S podem ser realizadas por uma grande variedade de dispositivos externos, também conhecidos como periféricos. Um dos dispositivos externos mais conhecidos, poderíamos dar como exemplo o teclado, que Stallings classifica como inteligíveis ao ser humano. Os classificados como inteligíveis à máquina serão os mais adequados, pois estes são compatíveis para comunicação com equipamentos. O dispositivo E/S permitirá a troca de dados entre os dispositivos ou até mesmo com outro computador. O dispositivo em questão trata-se de um adaptador USB Serial RS232 que estará em comunicação com o Arduino.

Os principais comandos de E/S servem para executar instruções que farão com que o dispositivo realize algo. Estes comandos são definidos por Stallings como:

- Controle: usado para ativar um periférico, ou ativar a leitura da porta Serial.
- Teste: usado para efetuar testes de estado de um modulo E/S, verificando-o se está ligado ou disponível para uso.
- Leitura: fará com que o modulo E/S obtenha dos dados disponíveis para leitura no dispositivo.
- Escrita: faz com que o modulo E/S envie um pacote de dados e transmita-o por um periférico correspondente.

Neste caso estas instruções serão controladas pelo servidor Web, onde utilizando a linguagem de programação em PHP terá condições de trata-las recebendo e enviando dados à dispositivos externos do mini computador.

3.6 IMPLEMENTAÇÃO DE HARDWARE

A implementação da placa Arduino e configuração de seus componentes são para recolher dados que em algum instante serão enviados para o mini computador. Da mesma forma o mini computador enviará dados para a placa Arduino que os traduzirá em comandos para controlar o módulo de reles e/ou dispositivos. Estas informações também serão mostradas no Display LCD 2004, que estará o tempo todo ativo durante o seu funcionamento.

A leitura dos Watts é essencial para realizar os cálculos e produzir informações que permitirão ao sistema controlar seu funcionamento, e para que isso seja feito a combinação de dois componentes é indispensável sendo um deles o medidor de Amperes e o outro o medidor de tensão. O sistema precisará mensurar os watts adquiridos durante o período de carregamento da bateira e para realizar esta leitura deverá ser efetuado um cálculo simples que é a multiplicação da tensão pelos amperes. Desta forma será possível saber quantos watts estão sendo gerados e também consumidos. A figura 14 mostra a estrutura de maneira resumida do sistema.

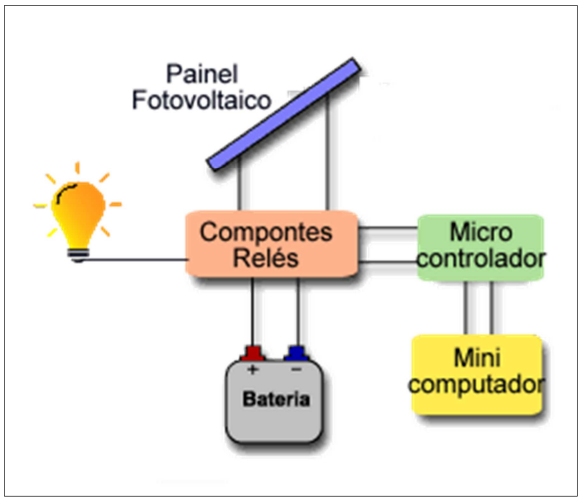


Figura 14 Estrutura resumida
Fonte: Do Autor

Seguindo a linha de raciocínio de tornar o sistema autônomo, claramente estamos dizendo que ele deve ser capaz de tomar decisões que o manterá em funcionando sem a interferência do componente humano. Como dito anteriormente o sistema faz leituras de consumo e geração de energia, então se o sistema perceber que o consumo está excedendo a geração de energia ele desligará todos os componentes externos ligados para se manter e apenas o microcontrolador e o mini computador ficarão ligados. Caso não haja uma nova recarga até que a bateria se esgote o sistema sofrera uma falha catastrófica onde o microcontrolador e o mini computador desligarão por falta de energia.

Uma medida de segurança que será feita de maneira permanente é a medição das tensões da bateira, caso a leitura destas tensões variem abaixo de 11,5 volts o sistema manterá todos os dispositivos externos desligados seguindo o mesmo princípio do consumo sob a geração de energia. O porquê de utilizar a tensão abaixo de 11,5 volts deve-se ao fato de que a tensão normal da bateria carregada se mantem acima de 12 volts, portanto ao mostrar valores menores que 12 volts a bateria estará descarregada necessitando ser reabastecida. Um

exemplo pode ser visto na figura 15, mostra parte do código que faz checagem na variável ‘vin’, esta variável recebe a leitura da porta analógica que mostra a tensão atual e se demonstrar valores que atendam à condição desligará o rele ativo e também a luz do LCD, tudo para economizar energia até que o sistema possa se reabastecer.

```
if(vin < 11.5){ // bateria descarregada - desliga lampada ou outros dispositivos
    digitalWrite(porta_relel, HIGH); // desliga o rele que estava ativando disositivo(s)
    timerbotao = 0; // zera timer para desligamento imediato
    lcd.setBacklight(LOW); // desliga iluminacao do LCD
}
```

Figura 15 Código checar tensão
Fonte: Do Autor

Estes processos de checagem serão muito importantes para a estabilidade do sistema, e tal controle vai depender das leituras realizadas pelos componentes e da programação criada para o sistema, estes terão efeito direto no desempenho. Perceber os efeitos de entrada de informações e retornar dados legíveis será à base da criação deste hardware e isto afetará na forma em como o sistema será utilizado.

3.7 SERVIDOR DE APLICATIVOS WEB PHP

3.7.1 Por que usar um servidor Web

A linguagem *Hypertext Transfer Protocol – HTTP*, é o idioma dos navegadores[11] presentes em nossos computadores pessoais, e através destes códigos os navegadores nos mostram conteúdos de forma que possamos absolver e interagir. Se criarmos um arquivo simples HTML[12] basta abri-lo com um navegador qualquer para visualizar as informações como uma página. Já o conceito Servidor Web trata-se de um computador que responde solicitações de outros computadores, o computador cliente utiliza um navegador para fazer requisições ao servidor sobre um determinado arquivo contido em seu repositório, o servidor web por sua vez processa a solicitação e responde ao requisitante em forma de HTML para que o navegador do cliente possa interpreta-lo. Este processo de requisição e resposta

[11] Um navegador é um programa interativo que permite ao usuário ver informações da World Wide Web. (Comer, 2007)
[12] HTML é a sigla de HyperText Markup Language, expressão inglesa que significa "Linguagem de Marcação de Hipertexto". Consiste em uma linguagem de marcação utilizada para produção de páginas na web, que permite a criação de documentos que podem ser lidos em praticamente qualquer tipo de computador e transmitidos pela internet. (Comer, 2007)

acontece por meio da conexão estabelecida entre cliente e servidor se utilizando de protocolos TCP.(COMER, 2007).

O HTTP torna fácil enviar informações utilizando a internet, e é claro que para utilizá-la devem-se estudar os riscos de segurança, mais esse não é o foco deste projeto. Este apenas está a desenvolver um sistema autônomo que captura sua própria energia de forma limpa e renovável e traz ainda outras funcionalidades como ligar lâmpadas que se utilizam da mesma fonte energética e ainda trará a possibilidade de controlar outros dispositivos, e estas informações serão apresentadas de forma simples utilizando um servidor Web. Outro fator importante que deve ser levado em conta é que existem softwares Open Source para a criação destas ferramentas, e basta apenas uma breve pesquisa na internet para achar tutoriais para configuração destes softwares. Analisando questões que serão necessárias para o sistema, deduzimos que não será possível trabalhar apenas com scripts HTML, e estudando opções e meios de comunicação entre o mini computador e o microcontrolador nasceu a necessidade de um código que interaja mais facilmente com o sistema operacional, e que trate de informações que serão enviadas ao computador pelo Arduino. Esta necessidade de interação deu-se a escolha de mais um software Open Source, e o eis que o PHP surgiu para fazer este papel.

Os arquivos interpretados por servidores Web, chamados de páginas HTML tem seu conceito também definido em diversos livros:

> Um documento hipermídia disponível na WEB é chamado de página; a página de uma organização ou de um indivíduo é conhecida como homepage. Para garantir que navegadores de Web possam interpretar corretamente os conteúdos de cada página, a Web usa uma representação padrão conhecida como *HypeText Markup Language (HTML)*. [...] porque dá apenas informações gerais para exibição, e não inclui instruções detalhadas de formatação. [...] permite a uma página especificar o nível de importância de um título, mas não requer que o autor especifique o tamanho do texto ou fonte exata a serem usados. [...] desta forma, a mesma página pode ser formatada para um monitor pequeno ou grande, uma exibição de alta ou baixa resolução, ou um dispositivo manual, como um PDA. COMER (2007, p480).

3.7.2 Criação do Servidor Web PHP

Nesta estapa 5 do fluxograma, fará o papel do servidor Web o Thin Client. Este mini computador adquirido em sucata eletrônica, foi descartado devido a um problema em seu pequeno disco. Para corrigir este problema o disco foi substituído por um pendrive de 8Gb que comportara uma pequena instalação de um sistema Open Source, o Ubuntu Server versão 10.04. A versão do Ubuntu Server 10.04 foi disponibilizada em abril de 2010, e seu período de suporte estendeu-se apenas até 2015, e isso afetou o repositório de softwares para instalação. Devido a isso sua lista de repositório (lista onde estão os endereços ou caminho

para instalação de Softwares) teve de ser atualizada. A figura 16 mostra os endereços editados na sources.list, e esses repositórios são mantidos pela comunidade de software livre espalhadas pela internet, o funcionamento deste repositório é muito importante pois facilita e muito a instalação do servidor Web Apache e pacotes para o funcionamento do PHP.

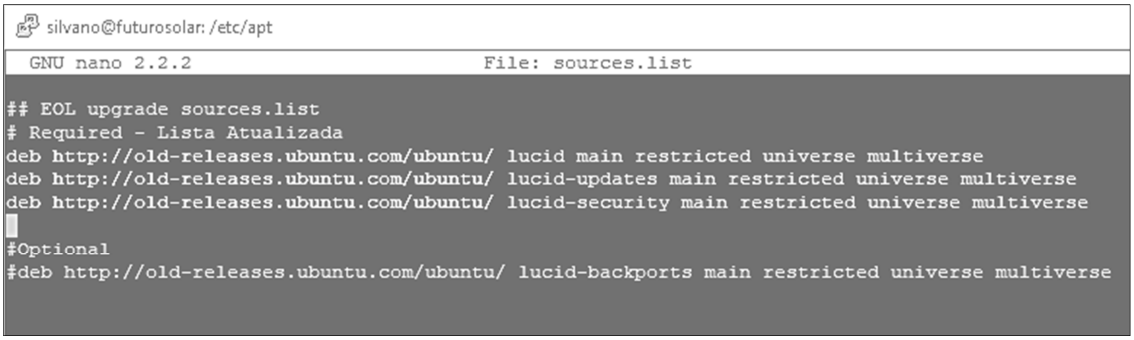


Figura 16 Sources.list
Fonte: Do Autor

Com o problema do repositório resolvido o mini computador Thin Client faz seu papel e rodará os softwares necessários, e mesmo com seus recursos bem limitados dará conta do recado. A instalação do software Apache Server poderá prosseguir e a partir dai hospedar nossa página PHP, e prosseguindo a instalação, a figura 17 mostra os comandos grifados update e install apache2 com seus pacotes PHP.

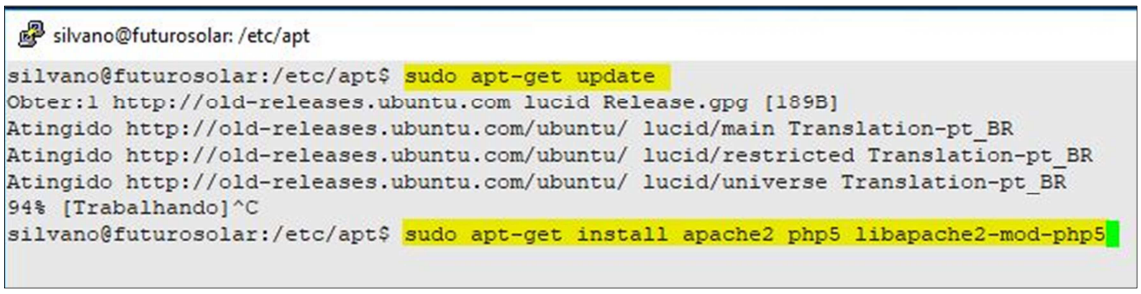


Figura 17 Comandos update e install
Fonte: Do Autor

Basta executar esses comandos e o Apache já estará disponível, assim será possível tratar informações diretamente dentro do servidor integrando HTML e PHP para criar uma interface compreensível para o usuário. Outras características do Thin Client que estarão disponíveis para uso são suas quatro portas USB, uma porta ethernet (Rede), uma porta VGA e uma saída de áudio. A comunicação com o Arduino estará sendo feita por um Adaptador USB RS232 enviando uma linha *string* com dados que serão capturados e tratados pelo PHP.

3.7.3 Funcionamento básico do PHP no Servidor Apache.

O PHP pode ser hospedado em diversos tipos de servidores compatíveis, mas como visto anteriormente neste projeto estamos utilizando o Apache Server, e sua adaptação com o Apache e quase que automática sem nenhuma dificuldade e dispensa outras instalações adicionais. O PHP e muito utilizado por iniciantes por conter uma gama enorme de exemplos pela internet.

Nesta última fase do fluxograma etapa 6, o PHP permitira visualizar e enviar as inrformações necessárias. Uma breve definição é demostrada de forma direta no conceito:

> O PHP é uma linguagem de criação de scripts do lado do servidor que foi projetada especificamente para a Web. Dentro de uma HTML, você pode embutir código PHP que será executado toda a vez que a página for visitada. O código PHP é interpretado pelo servidor Web e gera HTML ou outra saída que o visitante verá. [...] PHP é um produto de código-fonte aberto, o que significa que você tem acesso ao seu código fonte. WELLING & THOMSON, (2005, p.27)

O PHP é escrito misturando suas linhas de código em meio a HTML e por meio de tags identificamos onde começa e terminas seus códigos. A tag “<?php” é utilizada para diferenciar o código PHP do restante da página como mostra o exemplo na figura 18.

```
<?php
    // código PHP aqui
?>
```

Figura 18 Código PHP
Fonte: Do Autor

O HTML não tem muitos recursos para mudar seu formato de código e seu código tem um modelo engessado que praticamente nos obriga a copiar e colar todo ele em uma página, mas podemos deixar o código HTML como está e a partir daí criar as linhas de programação em PHP e o HTML vamos colando em todas as páginas que forem necessárias para o desenrolar do projeto. Um pequeno detalhe pode ser notado na extensão do arquivo que normalmente seria o (.html) e para executar os códigos corretamente passa a ser (.php) que será tratado pelo Apache como páginas de PHP. Na figura 19 podemos ver o código PHP inserido entre as tags do HTML.

```
<!DOCTYPE html>
<html >
    <head >
        <meta charset=”utf-8” >
        <meta name=”viewport” contente=”width=device-witdh”>
        <title> Pagina Teste </title>
    </head>
        <body>
            <?php
                / / código PHP aqui
            ?>
        </body>
</html>
```

Figura 19 Inserção do PHP
Fonte: Do Autor

Dentro das tags do PHP serão inseridas as linhas de códigos necessárias para recolher e tratar as informações que serão apresentadas na interface. Este servidor por ter recursos limitados, ter-se-á preocupação de não escrever códigos complexos e muito extensos, desta forma ficará mais fácil executa-las.

Segundo (Welling e Thomson, 2005, p.88) explica em sua metodologia que não é necessário se preocupar muito com o hardware ou ao software do servidor Web, pois uma das maiores qualidades do PHP é de funcionar muito bem em diversos ambientes, rodando desde os grandes servidores ou até mesmo em sistema muito menores. O PHP tem uma boa eficiência e pode atender milhões de acessos com um único servidor barato, superando o desempenho de outras linguagens. O processo de criação de *scripts* do lado do servidor irá facilitar criar conteúdos dinâmicos que serão disponibilizado ao usuário do site, e a medida que informações forem adquirida o conteúdo do site será alterado, e isso é o que o PHP faz de melhor. O PHP tem uma extensa biblioteca de funções e as chamando podemos utiliza-las para desenvolver aplicações Web, nesse ponto será utilizada uma *string* (dados de texto) para conseguirmos argumentos ou parâmetros da função. Esses argumentos serão utilizados nas funções para expressar resultados específicos.

Para responder as entradas de dados no código é preciso ser capaz de responder claramente para o usuário, e a programação terá de tomar decisões com referência nos dados obtidos da *string* recebida pela porta serial. A instrução *if* será utilizada para tomar as

decisões, e deverá fornecer a instrução *if* as condições a serem executas e estas deverão estar entre parênteses (). Um exemplo dado pelos autores Welling e Thomson (2005) (figura 20) mostra como pode ser organizado o código para que ele compare uma variável $totalqty verificando se seu valor é igual à zero:

```
if( $totalqty == 0 )
{
  echo 'You did not order anything on the previous page!<br />';
}
else
{
  echo $tireqty.' tires<br />';
```

Figura 20 Condição if
Fonte: Do Autor

Nota-se no código que após checar a condição a instrução *else* aparece e assume que uma ação alternativa seja aplicada, caso a condição do *if* não seja verdadeira, isso permite organizar o código e até mesmo chamar outra condição de checagem até que realmente exiba uma resposta satisfatória para a programação.

Ainda se baseando no estudo de Welling e Thomson (2005), podemos localizar trechos específicos de uma *string*, pois existem funções capazes de localizar uma parte especifica definida pelo programador. Modelos de implementação também podem ser encontrados no site oficial PHP.net. A função “strpos( )” tem o propósito de localizar algo dentro da *string* e permitir trabalha-la dentro de uma sub *string* “substr( )”, para assim poder capturar outros valores próximo ao conjunto de caracteres definidos na “strpos( )”, os autores definem que as funções retornam a posição numérica de uma agulha dentro de um palheiro e que o valor retornado representa a posição da ocorrência programada no código.

Com relação sobre a comunicação com a porta serial dentro do PHP adquirimos uma classe disponibilizada em código aberto e gratuita no fórum Arduino, desta forma o processo de leitura da serial dentro do sistema fica mais dinâmico necessitando apenas comandos de configuração da porta serial e leitura. A figura 21 mostra a parte do código responsável por esta configuração, isso ira garantir a criação da *string* que a programação do Arduino enviarpa na porta serial RS232. Essa linha *string* é armazenada na variável “$read” como mostrado na figura.

```php
<?php
    require("php_serial.class.php"); // inclui("php_serial.class.php");

    $serial = new phpSerial(); // lendo e startando a classe
    $serial->deviceSet("/dev/ttyUSB0"); // Linux serial device e /dev/ttyS0 para COM1, etc.

    $serial->confBaudRate(9600); //Baud rate: 9600serial
    $serial->confParity("none");  //Parity (this is the "N" in "8-N-1")
    $serial->confCharacterLength(8); //Character length     (this is the "8" in "8-N-1")
    $serial->confStopBits(1);  //Stop bits (this is the "1" in "8-N-1")
    $serial->confFlowControl("none");

    $serial->deviceOpen(); // abrindo conexao com serial
    while($read==''){
        $read = $serial->readPort(); // lendo os dados
        sleep(1);
    //$i++;
    }
```

Figura 21 Leitura Serial
Fonte: Do Autor

Após a variável $read estar armazenando a *string* já é possível usar a função strpos( ) para realizar a análise e recolher as informações para que as funções possam tomar as decisões de acordos com a programação do código. Nessa próxima imagem (figura 22) mostra a variável $posicao1 recebendo a localização da checagem do conjunto de caracteres (volts=) e após identificar onde dentro da string isso se faz verdadeiro a função substr( ) captura o valor contido após os valores numéricos definidos:

- substr($txtserial,$posicao1+6,5)

Esses valores (6,5) definem onde começam a capturar a informação e quantos caracteres serão capturados após o ponto de início.

```php
$txtserial = "$read";
$posicao1 = strpos($txtserial, 'volts='); // pega a string da saida da serial e seleciono a parte volts
$bater = substr($txtserial,$posicao1+6,5); // pega ex 12.00
```

Figura 22 Exemplo de código strpos( ) substr( )
Figura: Do Autor

3.7.4 Programando instruções *if* em PHP para representação de imagens.

Após a leitura da porta serial estar concluída, é possível trabalhar com uma *string* que possui os dados necessários para analise, e a criação de um pequeno sistema gráfico pode ser realizado, e construiremos o código PHP instruções *if* que trataram e armazenarão dados em variáveis que criarão este cenário.

Agora resta criar as instruções que tomarão as decisões e retornarão estes dados de forma que o usuário possa entender o que está sendo representado. Preocupado com essa interpretação foram criadas imagens que auxiliarão nesse processo e a figura 23 mostra a criação de seis imagens para representar a carga da bateria, onde a função analisará a tensão que foi recebida pela leitura dos dados enviados pela serial e mostrará uma figura correspondente com a checagem, e junto à figura imprimira o valor numérico indicando a porcentagem para a imagem apresentada.

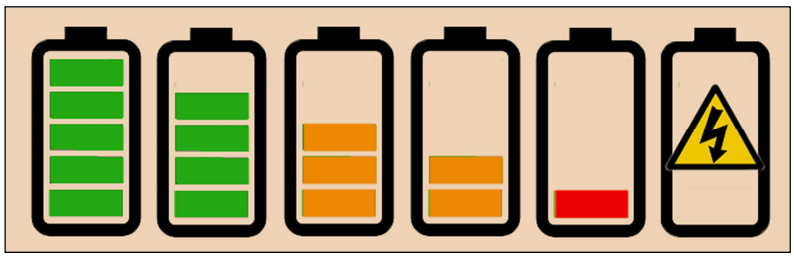


Figura 23 Percentual da bateria
Fonte: Do Autor

O código PHP é responsável por analisar e imprimir a imagem e o percentual de carga como demonstrado na figura 24.

```php
<?php
    if ($bateryF >= 13.1) { //condicao checando a tensao da bateria
            $imgbatery = "images/batery100.png";
        if($bateryF >= 13.4){
            $carga = "100%"; //subcondicao printa percentual 100% com a imagem
        }elseif($bateryF >= 13.1){
            $carga = "90%"; //subcondicao printa percentual 90% com a imagem
        }
    } elseif ($bateryF >= 12.5) {
            $imgbatery = "images/batery80.png";
        if($bateryF >= 12.8){
                        $carga = "80%";
                    }elseif($bateryF >= 12.5){
                        $carga = "70%";
                    }
    } elseif ($bateryF >= 11.9) {
                    $imgbatery = "images/batery60.png";
        if($bateryF >= 12.2){
                        $carga = "60%";
                    }elseif($bateryF >= 11.9){
                        $carga = "50%";
                    }
    } elseif ($bateryF >= 11.4) {
                    $imgbatery = "images/batery40.png";
        if($bateryF >= 11.6){
                        $carga = "40%";
                    }elseif($bateryF >= 11.4){
                        $carga = "30%";
                    }
     } elseif ($bateryF >= 11.0) {
                    $imgbatery = "images/batery20.png";
        if($bateryF >= 11.2){
                        $carga = "20%";
                    }elseif($bateryF >= 11.0){
                        $carga = "10%";
                    }
    } else { //se todas as anteriores forem falsas mostra bateria sem carga
        $imgbatery = "images/batery00.png";
```

Figura 24 Condições PHP Bateria
Fonte: Do Autor

Veja que o código começa com a *tag* do PHP e a partir dai as condições if entram em ação e tratam os valores recebidos. A variável $bateryF está mantendo as tenções que estão sendo lidas na *string* da serial, e nessa condição inicial é verificado se o valor da tensão recebida é maior ou igual a 13,1 volts, se essa condição for verdadeira o código armazena o endereço da imagem que corresponderá a carga atual na variável $imgbatery que será usado mais a frente na HTML para apresentar a imagem. Note que dentro desta mesma condição inicial existe outra condição que checa se a tensão e maior que 13,4 volts, senão for, checa se a tensão está entre 12,6 e 13,1 volts, assim o código armazenara em outra variável, sendo essa $carga o valor percentual que representara o status de carga da bateria.

Continuando neste mesmo sentido outras condições (elseif) são realizadas da mesma forma mais checando tensões diferentes, e se nenhumas das condições forem verdadeiras a variável $imgbatery recebera uma *string* com o caminho de uma imagem que corresponde que não há carga ou houve falha de leitura.

Outro conjunto de imagem que segue a mesma programação é mostrado na figura 25, refere-se à quantidade de carga que está sendo capturada pela placa solar e enviada para bateria. A diferença e que as variáveis trabalharão com os amperes lidos pelos componentes e assim representam o status atual da carga.

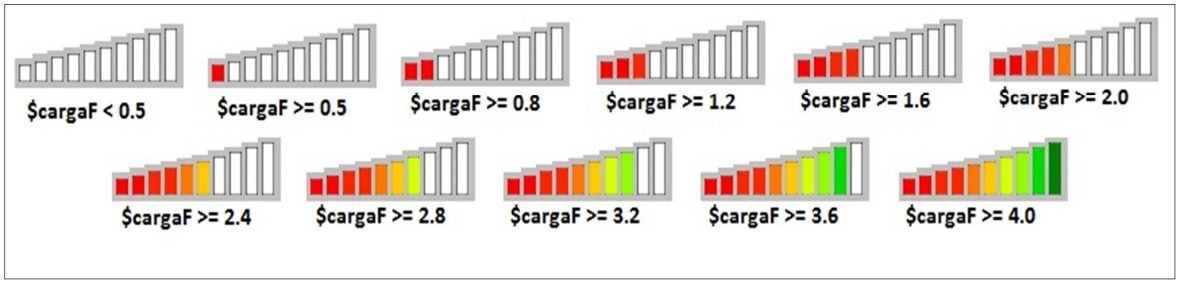


Figura 25 Corrente de carga
Fonte: Do Autor

Para finalizar, outras figuras são apresentadas na interface gráfica gerada com o servidor Web, mais todas seguem o mesmo raciocínio, e de forma bem resumida sem complicações no código à visualização final da interface real é representada na figura 26.

Figura 26 Página Web do Sistema
Fonte: Do Autor

3.7.5 Programando botões para ativar dispositivos com reles.

O HTML permite que sejam criados formulários que podem enviar comandos a uma outra página. Segundo (Welling e Thomson, 2005, p.4) a necessidade de se utilizar formulários vem para coletar informações do usuário. A obtenção de dados ou ações realizadas por um cliente ou usuário é muito mais fácil se trata-las dentro do PHP.

Note que na figura 26, existem cinco botões relacionados abaixo de um texto expressando um dispositivo, que neste caso trata-se de uma lâmpada, e cada um dos botões corresponde a um tempo que está definido na programação. Estes botões estão dentro de um formulário, que ao ser pressionado pelo usuário enviará dados à outra página PHP, que de acordo com a variável que lhe foi enviada, realizará uma ação específica e retornando um resultado na interface do sistema. Na figura 27 podemos ver parte do código do formulário e a *action* deste, que se trata da página correspondente ao qual o formulário faz a chamada para enviar os dados.

```
<!--------------POST DO BOTAO LIGAR 30 MIN -------------->
<form method="POST" action="arduino.php">
<p>
    <input type="hidden" value="30m" name="estado" >
    <input type="submit" value="Ligar 30 min" name="30min">
</p>
</form>
</td></tr><tr>

<!--------------POST DO BOTAO LIGAR 1 HORA -------------->

<form method="POST" action="arduino.php">

    <input type="hidden" value="1hora" name="estado" >
    <td><input type="submit" value="Ligar 1 Hora" name="1hora">
</td>
</form>
</tr><tr><td>

<!-------------POST DO BOTAO LIGAR 2 HORAS -------------->

<form method="POST" action="arduino.php">
<p>
    <input type="hidden" value="2horas" name="estado" >
    <input type="submit" value="Ligar 2 Horas" name="2horas">
</p>
</form>
</td></tr><tr>
```

Figura 27 Código formulário
Fonte: Do Autor

Após chamar a página arduino.php, é recebido a variável *string* que será tratada nas instruções desta página, e de acordo com o valor enviado pelo formulário será possível identificar qual foi a solicitação realizada pelo usuário na página do sistema. Esta por fim é responsável por interpretar a ação do usuário e informa-la para o arduino, assim o microcontrolador responde à ação ativando um dispositivo que foi solicitado pelo usuário, e neste caso o arduino ligará um rele, ativando uma lâmpada pelo tempo definido no botão selecionado na página Web do Sistema Autônomo.

4 CONCLUSÃO

No que se refere na realização deste projeto, pode-se concluir que o sistema autônomo alimentado por energia solar está concluido. Este sistema consiste em uma placa Arduino Uno, um módulo rele, um mini computador, uma placa fotovoltaica, componentes de medição de amperes, uma bateria de 12 volts 18 Ah e lâmpadas 12 volts. Com isso foi possível criar um sistema autônomo com um custo significativamente baixo, pois usou-se também conceito de reciclagem de lixo eletrônico, além de que a energia necessária para seu funcionamento é adquirida sob captação de energia solar, limpa e renovável.

O teste prático mostrou o sistema funcionando por mais de 30 dias sem a necessidade de intervenção humana, e a captação de energia solar foi eficaz. As ações humanas se destinaram apenas nas ativações de dispositivos controlados pela interface Web.

A realização deste trouxe um aprendizado de tecnologias que podem mudar a forma como entendermos o conforto em uma residência, e modernizar ambientes de forma que possamos interagir com os dispositivos presentes sem a necessidade de se locomover até eles, e ainda saber que a energia consumida por estes dispositivos é gratuita e renovável.

Por fim, o projeto superou as expectativas, apresentando um longo tempo de funcionamento e um bom desempenho mostrando que essa ideia está no caminho certo, motivando uma possível continuação futura do mesmo.

REFERÊNCIAS

AMERICADOSOL. Potencial solar no Brasil. Disponível em: <http://americadosol.org/potencial-solar-no-brasil#toggle-id-1>. Acesso em: 08 jan. 2017.

ARDUINO, The Oficial Web Site. Disponível em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 10 fev. 2017.

ARDUINO FORUM, Arduino and php serial class. Disponível em: <http://forum.arduino.cc/index.php?topic=54721.0>. Acesso em: 22 jan. 2017.

BARNABÉ, Moacir Luís; ROCHA, Rita de Cassia; VIEIRA, Carlos Eduardo Costa; SOUZA, Reginaldo Castro de. Tecnologia da Informação e Comunicação Modelo Computacional Baseado em Servidor: Estudo de Caso utilizando Thin Clients. 2010. Disponível em: <http://editorauss.uss.br/index.php/TECCEN/article/view/249/197>. Acesso em: 20 abr. 2017.

CIRCUITLAB: Edição de Circuitos Eletrônicos. Disponível em: <https://www.circuitlab.com/editor/>. Acesso em: 25 ago. 2017.

COELHO, PEDRO, Internet das Coisas – Introdução pratica, 1ª Edição. Lisboa: FCA – Editora de Informática Ltda. fevereiro de 2017.

COMER, Douglas E. Rede de Computadores e Internet. 4. ed. Porto Alegre: Bookman, 2007. 640 p.

LIMA, Charles Borges & VILLAÇA, Marco V. M. AVR e Arduino Técnicas de Projeto, 2ª edição. Florianópolis: Edição dos Autores 2012.

PEREIRA, Thulio Cícero Guimarães, Energias renováveis: políticas públicas e planejamento energético. Curitiba: COPEL, 2014.

PHP.NET. O que é o PHP. Disponível em: <https://secure.php.net/manual/pt_BR/intro-whatis.php>. Acesso em: 18 mar. 2017.

PHP.NET, Função substr: Retorna uma parte de uma string. Disponível em: <http://php.net/manual/pt_BR/function.substr.php>. Acesso em: 26 jan. 2017.

PINHO, João Tavares; GALDINO, Marco Antônio. Manual de Engenharia para Sistemas Fotovoltaicos. 2014. ed. Rio de Janeiro: CEPEL - CRESESB, 2014, p.530.

ROGER A. MESSENGER & JERRY VENTRE, Photovoltaic Systems Engineering, Third Edition, 2010.

SQUARESPACE: Datasheet LM7809. New Delhi, India. Disponível em: <https://static1.squarespace.com/static/5416a926e4b09de8832655bc/t/54427037e4b03de3b67b895a/1413640247188/lm7809.pdf>. Acesso em: 12 jun. 2017.

STALLINGS, WILLIAM, Arquitetura de Organização de Computadores. 10ª edição, São

Paulo: Pearson Education do Brasil, 2017.

TANENBAUM, AUSTIN, Organização estruturada de computadores. 6ª edição, São Paulo: Pearson Prentice Hall, 2013.

UOL, Folha de São Paulo. Número de smartphones em uso no Brasil chega a 168 milhões, diz estudo. 2016. Disponível em: <http://www1.folha.uol.com.br/mercado/2016/04/1761310-numero-de-smartphones-em-uso-no-brasil-chega-a-168-milhoes-diz-estudo.shtml>. Acesso em: 07 mar. 2018.

WELLING, Luke; THOMSON, Laura. PHP e MySQL Desenvolvimento Web. 2005. ed. Rio de Janeiro: Elsevier Editora Ltda, 2005. 754 p.

ANEXO I

1. Programa do Sistema – Microcontrolador

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,2,1,0,4,5,6,7,3,POSITIVE);

//Medidor de corrente amperes modelo ACS712 5A

int analogInputVc = A0;
int sensorIn2 = A2; //pino de entrada sensor Amperes consumido
int sensorIn3 = A3; //pino de entrada sensor Amperes gerados
int porta_rele1 = 5; //Porta ligada ao pino Sinal do modulo rele - lampada
int porta_rele2 = 6; //Porta ligada ao pino Sinal do modulo rele - Carga desliga
int porta_botao1 = 4; //Porta ligada ao botao 1
int porta_botao2 = 3; //Porta ligada ao botao 2
int value = 0;
float vout = 0.0;
float vin = 0.0; // variavel tensao volts
float R1 = 100000.0; // resistancia R1 (100K) !
float R2 = 10000.0; // resistancia R2 (10K) !
float maisWatts = 0.0; //variavel para soma em tempo real 1s
float menosWatts = 0.0; //variavel para soma em tempo real 1s
int leiturabt1 = 0; //Armazena o valor lido dos botoes
int leiturabt2 = 0; //Armazena o valor lido do botao desliga
int timerbotao = 0; //tempo em segundos do timer botao

float sensorValueAux2 = 0; //variavel auxiliar para aumentar precisao de leitura
float sensorValueAux3 = 0; //variavel auxiliar para aumentar precisao de leitura
float sensorValue2 = 0; //variavel necessaria para calculo sensor consumo
float sensorValue3 = 0; //variavel necessaria para calculo sensor de carregamento
float currentValue2 = 0; //variavel com resultador Amperes de consumo
float currentValue3 = 0; //variavel com resultador Amperes gerados no carregamento
float voltsporUnidade = 0.0048828125;

//time - simulador de tempo - variaveis
int segun = 0;
int minut = 0;
int hora = 0;
int dias = 0;
int tseg = 0;
int t2seg = 0;
int tmin = 0;

//variaveis para calcular watts hora
float convAmp2 = 0.0;
float minutoWatts2 = 0.0;
float horaWC = 0.0;
```

```
float convAmp3 = 0.0;
float minutoWatts3 = 0.0;
float horaWG = 0.0;

void setup(){
  Serial.begin(9600); // Inicializa a comunicação serial com uma taxa de 9600 bauds
  pinMode(analogInputVc, INPUT);
  pinMode(sensorIn2, INPUT);
  pinMode(sensorIn3, INPUT);
  lcd.begin(20,4);
  pinMode(porta_rele1, OUTPUT); //Define pinos para o rele como saida
  pinMode(porta_rele2, OUTPUT); //Define pinos para o rele como saida
  pinMode(porta_botao1, INPUT); //Define pinos dos botoes como entrada
  pinMode(porta_botao2, INPUT); //Define pinos dos botoes como entrada
}

void loop(){

  value = analogRead(analogInputVc);
  vout = (value * 5.0) / 1024.0; // 5v maximo porta arduino - entao sensor * 5.0
  vin = vout / (R2/(R1+R2));
  if (vin<0.09) {
  vin=0.0; //declaracao que elimina restos indesejados
  }
  if(vin > 13.8){
     digitalWrite(porta_rele2, LOW); // desliga o carregamento da bateria - corrente muito alta
e/ou bateria carregada
  }else if(vin < 13.3){
     digitalWrite(porta_rele2, HIGH); // liga o carregamento da bateria - corrente baixando -
mantendo a bateria carregada
  }
  if(vin < 11.5){ // bateria descarregada - desliga lampada ou outros dispositivos
     digitalWrite(porta_rele1, HIGH); // desliga o rele que estava ativando disositivo(s)
     timerbotao = 0; // zera timer para desligamento imediato
     lcd.setBacklight(LOW); // desliga iluminacao do LCD
  }
  for(int i=200; i>0; i--){ //efetuando 200 leituras para melhorar precisao do resultado
   sensorValueAux2 = (analogRead(sensorIn2) -511); //lendo o pino analogico A2
   sensorValue2 += pow(sensorValueAux2,2); //soma as leituras
  }
  for(int i=200; i>0; i--){
   sensorValueAux3 = (analogRead(sensorIn3) -511); //lendo o pino analogico A3
   sensorValue3 += pow(sensorValueAux3,2); //soma as leituras
  }

  sensorValue2 = (sqrt(sensorValue2/ 200)) * voltsporUnidade; //finaliza o valor dividindo
pela quantidade de vezes lida
  currentValue2 = (sensorValue2/185)*2500; //calcula a corrente com sensibilidade do sernsor
185 mV para 5A (5000mha/2)=2500
```

```
  sensorValue3 = (sqrt(sensorValue3/ 200)) * voltsporUnidade; //finaliza o valor dividindo pela quantidade de vezes lida
  currentValue3 = (sensorValue3/185)*2500; //calcula a corrente com sensibilidade do sernsor 185 mV para 5A

   /////////////// inicio do botao timer
   leiturabt1 = digitalRead(porta_botao1);
   leiturabt2 = digitalRead(porta_botao2);
   /////////////// leitura porta serial
   char caracter;
   caracter = Serial.read();
   //se pressionado "e" liga 30m
   if(caracter == 'e')
   {
     timerbotao=30;
     digitalWrite(porta_rele1, LOW);
     //lcd.setBacklight(HIGH);
     //displayon = 60; //Acende display por 60 segundos quando precionado botao
   }
   else
    //Se pressionado "f" liga 1 hora
    if(caracter == 'f')
    {
       timerbotao=60;
       digitalWrite(porta_rele1, LOW);
       //lcd.setBacklight(HIGH);
       //displayon = 60; //Acende display por 60 segundos quando precionado botao
    }
    else
      //S pressionado "g" liga 2 horas
      if(caracter == 'g')
      {
        timerbotao=120;
        digitalWrite(porta_rele1, LOW);
        //lcd.setBacklight(HIGH);
        //displayon = 60; //Acende display por 60 segundos quando precionado botao
      }
      else
        //Se pressionado "j" liga 3 horas
        if(caracter == 'j')
        {
          timerbotao=180;
          digitalWrite(porta_rele1, LOW);
          //lcd.setBacklight(HIGH);
          //displayon = 60; //Acende display por 60 segundos quando precionado botao
        }
        else
          //Se pressionado "d" desliga
          if(caracter == 'd')
          {
```

```
            timerbotao=0;
            digitalWrite(porta_rele1, HIGH);
            //lcd.setBacklight(HIGH);
            //displayon = 60; //Acende display por 60 segundos quando precionado botao
          }
  if (leiturabt1 != 0) //Acao do botao 1 e adicionar 30 minutos no timerbotao e ligar rele das
lapadas
    {
     timerbotao = timerbotao+30;
     digitalWrite(porta_rele1, LOW);
     //lcd.setBacklight(HIGH);
     //displayon = 60; //Acende display por 60 segundos quando precionado botao
    }
    if (timerbotao == 0 ){
     digitalWrite(porta_rele1, HIGH);
    }
    if (leiturabt2 != 0) //Acao do botao 2 e zerar timer e deligar o rele das lampadas
    {
     timerbotao = 0;
     digitalWrite(porta_rele1, HIGH);
     //lcd.setBacklight(HIGH);
     //displayon = 60; //Acende display por 60 segundos quando precionado botao
    }

 segun++; // incrementa o tempo em segundos
 convAmp2 = currentValue2 * vin;
 minutoWatts2 = minutoWatts2 + convAmp2;
 convAmp3 = currentValue3 * vin;
 minutoWatts3 = minutoWatts3 + convAmp3;
 if(segun == 60){
    minut++;
    timerbotao--; //variavel tempo para desligar rele das lampadas
    tseg = segun;
    segun = 0;
    if(timerbotao <= 0){ //Pequena funcao para nao deixar valor timerbotao negativo
      timerbotao = 0;
    }
    ////////////////////////////////////////////////
    horaWC = horaWC + (minutoWatts2 / 3600);
    horaWG = horaWG + (minutoWatts3 / 3600);
    minutoWatts2 = 0; //zera variavel que acumula watts hora
    minutoWatts3 = 0; //zera variavel que acumula watts hora
    ////////////////////////////////////////////////
    if(minut == 60 && tseg == 60){
        hora++;
        tmin = minut;
        minut = 0;
        t2seg = tseg;
        tseg = 0;
      if(hora == 24 && tmin == 60 && t2seg == 60){
```

```
          dias++;
          tmin = 0;
          t2seg = 0;
          hora = 0;
       }
     }
  }

//daqui imprime na serial
 Serial.print('\n');
 Serial.print('\r');
 Serial.print(" volts=");
 Serial.print(vin); // imprime voltagem na serial
 Serial.print(" WG=");
 Serial.print(horaWG,2); // imprime carga gerada na serial
 Serial.print(" WC=");
 Serial.print(horaWC,2); // imprime consumo na serial
 Serial.print(" Ah=");
 Serial.print(currentValue3,2); // imprime voltagem na serial
 Serial.print(" Li=");
 Serial.print(timerbotao); // imprime o tempo da luz acesa

// mostra o resultado no terminal
lcd.setCursor(1,0);
lcd.print("A-=" );
lcd.print(currentValue2,2);
lcd.setCursor(9,0);
lcd.print("WC=" );
lcd.print(horaWC,2);
lcd.setCursor(1,1);
lcd.print("A+=" );
lcd.print(currentValue3,2);
lcd.setCursor(9,1);
lcd.print("WG=" );
lcd.print(horaWG,2);
lcd.setCursor(1,2);
lcd.print("Volt = " );
lcd.print(vin,2);
//data e hora
lcd.setCursor(1,3);
lcd.print("            ");
lcd.setCursor(0,3);
lcd.print("T");
lcd.print(dias);
lcd.print(":");
lcd.print(hora);
lcd.print(":");
lcd.print(minut);
lcd.print(":");
lcd.print(segun);
```

```
 sensorValue2 =0;
 currentValue2 =0;
 sensorValue3 =0;
 currentValue3 =0;
 delay(860);
}
```

ANEXO II

1. Pagina da Web – index.php

```
<html>
<?php
$data = shell_exec('uptime'); // mostra tempo do servido - ligado
$uptime = explode(' up ', $data);
$uptime = explode(',', $uptime[1]);
$uptime = $uptime[0].', ';//.$uptime[1]; //o array comentado $uptime[1] mosta qtd usuarios do sistema logados
?>

<?php
   require("php_serial.class.php"); // inclui("php_serial.class.php");

   $serial = new phpSerial; // lendo e startando a classe
   $serial->deviceSet("/dev/ttyUSB0"); // Linux serial device e /dev/ttyS0 para COM1, etc.
   $serial->confBaudRate(9600); //Baud rate: 9600 serial
   $serial->confParity("none");  //Parity (this is the "N" in "8-N-1")
   $serial->confCharacterLength(8); //Character length     (this is the "8" in "8-N-1")
   $serial->confStopBits(1);  //Stop bits (this is the "1" in "8-N-1")
   $serial->confFlowControl("none");

   $serial->deviceOpen(); // abrindo conexao com serial
   while($read==''){
         $read = $serial->readPort(); // lendo os dados
         sleep(2);
   }

$txtserial = "$read";
$posicao1 = strpos($txtserial, 'volts='); // pega a string da saida da serial e seleciono a parte volts
$bater = substr($txtserial,$posicao1+6,5); // pega ex 12.00
$posicao2 = strpos($txtserial, 'Ah='); // pega a string da saida da serial e seleciono a parte amperes
$carga = substr($txtserial,$posicao2+3,4); // pega ex 1.02
$posicao3 = strpos($txtserial, 'WG='); // pega a string da saida da serial e seleciono a parte watts gerado
$wattger = substr($txtserial,$posicao3+3,5); // pega ex 1.02
$posicao4 = strpos($txtserial, 'WC='); // pega a string da saida da serial e seleciono a parte watts consumido
$wattcon = substr($txtserial,$posicao4+3,5); // pega ex 1.02
$light = strpos($txtserial, 'Li=');
$lighton = substr($txtserial,$light+3,2);

//$text = "$read";
//$bater = substr($text, 7, 5);
$bateryF = (float) $bater; //convertendo o valor capturado da string para float
$cargaF = (float) $carga; //convertendo o valor para float
$wattgerI = (int) $wattger; //convertendo o valor da carga para inteiro
```

```
$wattconI = (int) $wattcon; //convertendo o valor do consumo para inteiro


// Gerando dados percentual consumo
$percTotal = $wattgerI + $wattconI;
$percGerac = ($wattgerI / $percTotal)*100;
$percConsu = ($wattconI / $percTotal)*100;
$pGera = (int) $percGerac;
$pCons = (int) $percConsu;

// Gerador de grafico  CHART GOOGLE
        function geraGrafico($largura, $altura, $valores, $referencias, $tipo = "p3"){
        $valores = implode(',', $valores);
        $referencias = implode('|', $referencias);
        return "http://chart.apis.google.com/chart?chs=". $largura ."x". $altura .
"&amp;chd=t:" . $valores . "&amp;cht=p3&amp;chl=" . $referencias;
    }
$grafico = geraGrafico(305, 100, array("$pGera", "$pCons"), array("Watts Gerad", "Watts
Cons"))

?>

        <head>
                <meta http-equiv="refresh" content="30;URL=http://190.168.120.120">
                <title> Sistema E-SOLAR </title>
                <meta name="description" content="Sistema Servidor autonomo com energia
solar!! ">
        </head>

        <body>
                <br /><font face="Georgia" size="4"> <?php echo ('Tempo sistema Online - ');
?><font size="8"><?php echo (".$uptime."); ?> </font><img src="images/energ-solar.png"
width=150 height=150 <br />
                <br /><img src="images/seta-renovavel.gif" width=80 height=80 >
                <?php
                if ($lighton <= 0){
                        $imgluz = "images/lightoff.png";
                        $lux = "Off";
                } else {
                        $imgluz = "images/lighton.png";
                        $lux = "ON";
                }
                ?>
                <img src=<?php print_r($imgluz) ?> width=80 height=80 >
                <br /><br /><img src="images/link-rede.gif" width=20 height=20 >
                <!--
                //Printado a saida da serial
                -->
                <br /><br /><font face="Georgia" size="4">RS232: <?php print_r($read);
$serial->deviceClose(); ?><br />
```

```
<?php
        if ($bateryF >= 13.1) {
                $imgbatery = "images/batery100.png";
                if($bateryF >= 13.4){
                        $carga = "100%";
                }elseif($bateryF >= 13.1){
                        $carga = "90%";
                }
        } elseif ($bateryF >= 12.5) {
                $imgbatery = "images/batery80.png";
                if($bateryF >= 12.8){
            $carga = "80%";
       }elseif($bateryF >= 12.5){
            $carga = "70%";
       }
        } elseif ($bateryF >= 11.9) {
       $imgbatery = "images/batery60.png";
                if($bateryF >= 12.2){
            $carga = "60%";
       }elseif($bateryF >= 11.9){
            $carga = "50%";
       }
        } elseif ($bateryF >= 11.4) {
       $imgbatery = "images/batery40.png";
                if($bateryF >= 11.6){
            $carga = "40%";
       }elseif($bateryF >= 11.4){
            $carga = "30%";
       }
         } elseif ($bateryF >= 11.0) {
       $imgbatery = "images/batery20.png";
                if($bateryF >= 11.2){
            $carga = "20%";
       }elseif($bateryF >= 11.0){
            $carga = "10%";
       }
        } else {
                $imgbatery = "images/batery00.png";
                $carga = "1%";
        }
        // criando a imagem da carga corrente
        if ($cargaF >= 4.0){
                $imgcarga = "images/ah-carga100.png";
                $carregando = "MAX";
        } elseif ($cargaF >= 3.6){
       $imgcarga = "images/ah-carga90.png";
       $carregando = "90%";
        } elseif ($cargaF >= 3.2){
       $imgcarga = "images/ah-carga80.png";
       $carregando = "80%";
```

```
                    } elseif ($cargaF >= 2.8){
                $imgcarga = "images/ah-carga70.png";
                $carregando = "70%";
                    } elseif ($cargaF >= 2.4){
                $imgcarga = "images/ah-carga60.png";
                $carregando = "60%";
                    } elseif ($cargaF >= 2.0){
                $imgcarga = "images/ah-carga50.png";
                $carregando = "50%";
                    } elseif ($cargaF >= 1.6){
                $imgcarga = "images/ah-carga40.png";
                $carregando = "40%";
                    } elseif ($cargaF >= 1.2){
                $imgcarga = "images/ah-carga30.png";
                $carregando = "30%";
                    } elseif ($cargaF >= 0.8){
                $imgcarga = "images/ah-carga20.png";
                $carregando = "20%";
                    } elseif ($cargaF >= 0.5){
                $imgcarga = "images/ah-carga10.png";
                $carregando = "10%";
                    } else {
                $imgcarga = "images/ah-carga.png";
                $carregando = "S/Carga";
                    }
            ?><br />
            <?php $percent = (($wattgerI/$wattconI)*100);  //gerando o percentual da
carga sobre consumo
            ?>
            <table border=0 width='50%'><tr><td><img src=<?php print_r($imgbatery);
?> width=50 height=100 ></td><td><font face="Georgia" size="3">eficiencia/carga<br
/><img src=<?php print_r($imgcarga); ?> width=130 height=50 ></td><td><img src="<?php
echo $grafico ?>" title="Consumo!" /></td><br />
            <tr><td><?php print_r($carga); ?></td><td><?php print_r($carregando);
?></td><td><font face="Georgia" size="3"><?php echo $wattgerI ?>/<?php echo $wattconI
?> __<?php echo number_format($percent, 2, ".", ""); ?>% </td></tr></table><br />

        <!-- ###################### FORMULARIOS ###################### -->
        <h2>LAMPADA 01</h2>
        <table border="0" width="15%" cellpadding="0">
      <tr><td>
        <!--------------POST DO BOTAO LIGAR 30 MIN -------------->
        <form method="POST" action="arduino.php">
            <input type="hidden" value="30min" name="estado" >
            <input type="submit" value="Ligar 30 min" name="30min">
        </form>
        </td></tr><tr><td>
        <!--------------POST DO BOTAO LIGAR 1 HORA -------------->
        <form method="POST" action="arduino.php">
            <input type="hidden" value="1hora" name="estado" >
```

```
                <input type="submit" value="Ligar 1 Hora" name="1hora">
            </td>
            </form>
        </tr><tr><td>
            <!-------------POST DO BOTAO LIGAR 2 HORAS -------------->
            <form method="POST" action="arduino.php">
                <input type="hidden" value="2horas" name="estado" >
                <input type="submit" value="Ligar 2 Horas" name="2horas">
            </form>
        </td></tr><tr><td>
            <!---------- POST DO BOTAO LIGAR 3 HORAS ---------------->
            <form method="POST" action="arduino.php">
                <input type="hidden" value="3horas" name="estado" >
                <input type="submit" value="Ligar 3 Horas" name="3horas">
            </form>
        </td></tr><tr>
            <!--------- POST DO BOTAO DESLIGAR ---------------------->
            <form method="POST" action="arduino.php">
                <input type="hidden" value="Desl" name="estado" >
                <td><input type="submit" value="DESLIGAR" name="Desl">
            </td>
            </form>
            </tr>
            </table>
            </div>
            </body>
</html>
```

2. Classe serial php_serial.class.php

```
<?php
define ("SERIAL_DEVICE_NOTSET", 0);
define ("SERIAL_DEVICE_SET", 1);
define ("SERIAL_DEVICE_OPENED", 2);

/**
 * Serial port control class
 *
 * THIS PROGRAM COMES WITH ABSOLUTELY NO WARANTIES !
 * USE IT AT YOUR OWN RISKS !
 *
 * @author Rémy Sanchez <thenux@gmail.com>
 * @thanks Aurélien Derouineau for finding how to open serial ports with windows
 * @thanks Alec Avedisyan for help and testing with reading
 * @copyright under GPL 2 licence
 */
class phpSerial
{
        var $_device = null;
```

```
        var $_windevice = null;
        var $_dHandle = null;
        var $_dState = SERIAL_DEVICE_NOTSET;
        var $_buffer = "";
        var $_os = "";

        /**
         * This var says if buffer should be flushed by sendMessage (true) or manualy (false)
         *
         * @var bool
         */
        var $autoflush = true;

        /**
         * Constructor. Perform some checks about the OS and setserial
         *
         * @return phpSerial
         */
        function phpSerial ()
        {
                setlocale(LC_ALL, "en_US");

                $sysname = php_uname();

                if (substr($sysname, 0, 5) === "Linux")
                {
                        $this->_os = "linux";

                        if($this->_exec("stty --version") === 0)
                        {
                                register_shutdown_function(array($this, "deviceClose"));
                        }
                        else
                        {
                                trigger_error("No stty availible, unable to run.",
E_USER_ERROR);
                        }
                }
                elseif(substr($sysname, 0, 7) === "Windows")
                {
                        $this->_os = "windows";
                        register_shutdown_function(array($this, "deviceClose"));
                }
                else
                {
                        trigger_error("Host OS is neither linux nor windows, unable tu run.",
E_USER_ERROR);
                        exit();
                }
        }
```

```
        //
        // OPEN/CLOSE DEVICE SECTION -- {START}
        //

        /**
         * Device set function : used to set the device name/address.
         * -> linux : use the device address, like /dev/ttyS0
         * -> windows : use the COMxx device name, like COM1 (can also be used
         *     with linux)
         *
         * @param string $device the name of the device to be used
         * @return bool
         */
        function deviceSet ($device)
        {
                if ($this->_dState !== SERIAL_DEVICE_OPENED)
                {
                        if ($this->_os === "linux")
                        {
                                if (preg_match("@^COM(\d+):?$@i", $device, $matches))
                                {
                                        $device = "/dev/ttyS" . ($matches[1] - 1);
                                }

                                if ($this->_exec("stty -F " . $device) === 0)
                                {
                                        $this->_device = $device;
                                        $this->_dState = SERIAL_DEVICE_SET;
                                        return true;
                                }
                        }
                        elseif ($this->_os === "windows")
                        {
                                if (preg_match("@^COM(\d+):?$@i", $device, $matches) and
$this->_exec(exec("mode " . $device)) === 0)
                                {
                                        $this->_windevice = "COM" . $matches[1];
                                        $this->_device = "\\.\com" . $matches[1];
                                        $this->_dState = SERIAL_DEVICE_SET;
                                        return true;
                                }
                        }

                        trigger_error("Specified serial port is not valid",
E_USER_WARNING);
                        return false;
                }
                else
                {
```

```
                trigger_error("You must close your device before to set an other one",
E_USER_WARNING);
                return false;
            }
        }

        /**
         * Opens the device for reading and/or writing.
         *
         * @param string $mode Opening mode : same parameter as fopen()
         * @return bool
         */
        function deviceOpen ($mode = "r+b")
        {
            if ($this->_dState === SERIAL_DEVICE_OPENED)
            {
                trigger_error("The device is already opened", E_USER_NOTICE);
                return true;
            }

            if ($this->_dState === SERIAL_DEVICE_NOTSET)
            {
                trigger_error("The device must be set before to be open",
E_USER_WARNING);
                return false;
            }

            if (!preg_match("@^[raw]\+?b?$@", $mode))
            {
                trigger_error("Invalid opening mode : ".$mode.". Use fopen() modes.",
E_USER_WARNING);
                return false;
            }

            $this->_dHandle = @fopen($this->_device, $mode);

            if ($this->_dHandle !== false)
            {
                stream_set_blocking($this->_dHandle, 0);
                $this->_dState = SERIAL_DEVICE_OPENED;
                return true;
            }

            $this->_dHandle = null;
            trigger_error("Unable to open the device", E_USER_WARNING);
            return false;
        }

        /**
         * Closes the device
```

```
     *
     * @return bool
     */
    function deviceClose ()
    {
        if ($this->_dState !== SERIAL_DEVICE_OPENED)
        {
            return true;
        }

        if (fclose($this->_dHandle))
        {
            $this->_dHandle = null;
            $this->_dState = SERIAL_DEVICE_SET;
            return true;
        }

        trigger_error("Unable to close the device", E_USER_ERROR);
        return false;
    }

    //
    // OPEN/CLOSE DEVICE SECTION -- {STOP}
    //

    //
    // CONFIGURE SECTION -- {START}
    //

    /**
     * Configure the Baud Rate
     * Possible rates : 110, 150, 300, 600, 1200, 2400, 4800, 9600, 38400,
     * 57600 and 115200.
     *
     * @param int $rate the rate to set the port in
     * @return bool
     */
    function confBaudRate ($rate)
    {
        if ($this->_dState !== SERIAL_DEVICE_SET)
        {
            trigger_error("Unable to set the baud rate : the device is either not set or
opened", E_USER_WARNING);
            return false;
        }

        $validBauds = array (
            110    => 11,
            150    => 15,
            300    => 30,
```

```
                        600    => 60,
                        1200   => 12,
                        2400   => 24,
                        4800   => 48,
                        9600   => 96,
                        19200  => 19,
                        38400  => 38400,
                        57600  => 57600,
                        115200 => 115200
                );

                if (isset($validBauds[$rate]))
                {
                        if ($this->_os === "linux")
                        {
                                $ret = $this->_exec("stty -F " . $this->_device . " " . (int) $rate,
$out);
                        }
                        elseif ($this->_os === "windows")
                        {
                                $ret = $this->_exec("mode " . $this->_windevice . " BAUD=" .
$validBauds[$rate], $out);
                        }
                        else return false;

                        if ($ret !== 0)
                        {
                                trigger_error ("Unable to set baud rate: " . $out[1],
E_USER_WARNING);
                                return false;
                        }
                }
        }

        /**
         * Configure parity.
         * Modes : odd, even, none
         *
         * @param string $parity one of the modes
         * @return bool
         */
        function confParity ($parity)
        {
                if ($this->_dState !== SERIAL_DEVICE_SET)
                {
                        trigger_error("Unable to set parity : the device is either not set or
opened", E_USER_WARNING);
                        return false;
                }
```

```
                $args = array(
                        "none" => "-parenb",
                        "odd"  => "parenb parodd",
                        "even" => "parenb -parodd",
                );

                if (!isset($args[$parity]))
                {
                        trigger_error("Parity mode not supported", E_USER_WARNING);
                        return false;
                }

                if ($this->_os === "linux")
                {
                        $ret = $this->_exec("stty -F " . $this->_device . " " . $args[$parity],
$out);
                }
                else
                {
                        $ret = $this->_exec("mode " . $this->_windevice . " PARITY=" .
$parity{0}, $out);
                }

                if ($ret === 0)
                {
                        return true;
                }

                trigger_error("Unable to set parity : " . $out[1], E_USER_WARNING);
                return false;
        }

        /**
         * Sets the length of a character.
         *
         * @param int $int length of a character (5 <= length <= 8)
         * @return bool
         */
        function confCharacterLength ($int)
        {
                if ($this->_dState !== SERIAL_DEVICE_SET)
                {
                        trigger_error("Unable to set length of a character : the device is either
not set or opened", E_USER_WARNING);
                        return false;
                }

                $int = (int) $int;
                if ($int < 5) $int = 5;
                elseif ($int > 8) $int = 8;
```

```
                if ($this->_os === "linux")
                {
                        $ret = $this->_exec("stty -F " . $this->_device . " cs" . $int, $out);
                }
                else
                {
                        $ret = $this->_exec("mode " . $this->_windevice . " DATA=" . $int,
$out);
                }

                if ($ret === 0)
                {
                        return true;
                }

                trigger_error("Unable to set character length : " .$out[1],
E_USER_WARNING);
                return false;
        }

        /**
         * Sets the length of stop bits.
         *
         * @param float $length the length of a stop bit. It must be either 1,
         * 1.5 or 2. 1.5 is not supported under linux and on some computers.
         * @return bool
         */
        function confStopBits ($length)
        {
                if ($this->_dState !== SERIAL_DEVICE_SET)
                {
                        trigger_error("Unable to set the length of a stop bit : the device is either
not set or opened", E_USER_WARNING);
                        return false;
                }

                if ($length != 1 and $length != 2 and $length != 1.5 and !($length == 1.5 and
$this->_os === "linux"))
                {
                        trigger_error("Specified stop bit length is invalid",
E_USER_WARNING);
                        return false;
                }

                if ($this->_os === "linux")
                {
                        $ret = $this->_exec("stty -F " . $this->_device . " " . (($length == 1) ? "-
" : "") . "cstopb", $out);
                }
```

```
                else
                {
                        $ret = $this->_exec("mode " . $this->_windevice . " STOP=" . $length,
$out);
                }

                if ($ret === 0)
                {
                        return true;
                }

                trigger_error("Unable to set stop bit length : " . $out[1],
E_USER_WARNING);
                return false;
        }

        /**
         * Configures the flow control
         *
         * @param string $mode Set the flow control mode. Availible modes :
         *      -> "none" : no flow control
         *      -> "rts/cts" : use RTS/CTS handshaking
         *      -> "xon/xoff" : use XON/XOFF protocol
         * @return bool
         */
        function confFlowControl ($mode)
        {
                if ($this->_dState !== SERIAL_DEVICE_SET)
                {
                        trigger_error("Unable to set flow control mode : the device is either not
set or opened", E_USER_WARNING);
                        return false;
                }

                $linuxModes = array(
                        "none"     => "clocal -crtscts -ixon -ixoff",
                        "rts/cts"  => "-clocal crtscts -ixon -ixoff",
                        "xon/xoff" => "-clocal -crtscts ixon ixoff"
                );
                $windowsModes = array(
                        "none"     => "xon=off octs=off rts=on",
                        "rts/cts"  => "xon=off octs=on rts=hs",
                        "xon/xoff" => "xon=on octs=off rts=on",
                );

                if ($mode !== "none" and $mode !== "rts/cts" and $mode !== "xon/xoff") {
                        trigger_error("Invalid flow control mode specified",
E_USER_ERROR);
                        return false;
                }
```

```
                if ($this->_os === "linux")
                        $ret = $this->_exec("stty -F " . $this->_device . " " .
$linuxModes[$mode], $out);
                else
                        $ret = $this->_exec("mode " . $this->_windevice . " " .
$windowsModes[$mode], $out);

                if ($ret === 0) return true;
                else {
                        trigger_error("Unable to set flow control : " . $out[1],
E_USER_ERROR);
                        return false;
                }
        }

        /**
         * Sets a setserial parameter (cf man setserial)
         * NO MORE USEFUL !
         *       -> No longer supported
         *       -> Only use it if you need it
         *
         * @param string $param parameter name
         * @param string $arg parameter value
         * @return bool
         */
        function setSetserialFlag ($param, $arg = "")
        {
                if (!$this->_ckOpened()) return false;

                $return = exec ("setserial " . $this->_device . " " . $param . " " . $arg . "
2>&1");

                if ($return{0} === "I")
                {
                        trigger_error("setserial: Invalid flag", E_USER_WARNING);
                        return false;
                }
                elseif ($return{0} === "/")
                {
                        trigger_error("setserial: Error with device file", E_USER_WARNING);
                        return false;
                }
                else
                {
                        return true;
                }
        }

        //
```

```
        // CONFIGURE SECTION -- {STOP}
        //

        //
        // I/O SECTION -- {START}
        //

        /**
         * Sends a string to the device
         *
         * @param string $str string to be sent to the device
         * @param float $waitForReply time to wait for the reply (in seconds)
         */
        function sendMessage ($str, $waitForReply = 0.1)
        {
                $this->_buffer .= $str;

                if ($this->autoflush === true) $this->flush();

                usleep((int) ($waitForReply * 1000000));
        }

        /**
         * Reads the port until no new datas are availible, then return the content.
         *
         * @pararm int $count number of characters to be read (will stop before
         *        if less characters are in the buffer)
         * @return string
         */
        function readPort ($count = 0)
        {
                if ($this->_dState !== SERIAL_DEVICE_OPENED)
                {
                        trigger_error("Device must be opened to read it",
E_USER_WARNING);
                        return false;
                }

                if ($this->_os === "linux")
                {
                        $content = ""; $i = 0;

                        if ($count !== 0)
                        {
                                do {
                                        if ($i > $count) $content .= fread($this->_dHandle,
($count - $i));
                                        else $content .= fread($this->_dHandle, 128);
                                } while (($i += 128) === strlen($content));
                        }
```

```
                else
                {
                        do {
                                $content .= fread($this->_dHandle, 128);
                        } while (($i += 128) === strlen($content));
                }

                return $content;
        }
        elseif ($this->_os === "windows")
        {
                /* Do nohting : not implented yet */
        }

        trigger_error("Reading serial port is not implemented for Windows",
E_USER_WARNING);
        return false;
    }

    /**
     * Flushes the output buffer
     *
     * @return bool
     */
    function flush ()
    {
        if (!$this->_ckOpened()) return false;

        if (fwrite($this->_dHandle, $this->_buffer) !== false)
        {
            $this->_buffer = "";
            return true;
        }
        else
        {
            $this->_buffer = "";
            trigger_error("Error while sending message", E_USER_WARNING);
            return false;
        }
    }

    //
    // I/O SECTION -- {STOP}
    //

    //
    // INTERNAL TOOLKIT -- {START}
    //

    function _ckOpened()
```

```
        {
                if ($this->_dState !== SERIAL_DEVICE_OPENED)
                {
                        trigger_error("Device must be opened", E_USER_WARNING);
                        return false;
                }

                return true;
        }

        function _ckClosed()
        {
                if ($this->_dState !== SERIAL_DEVICE_CLOSED)
                {
                        trigger_error("Device must be closed", E_USER_WARNING);
                        return false;
                }

                return true;
        }

        function _exec($cmd, &$out = null)
        {
                $desc = array(
                        1 => array("pipe", "w"),
                        2 => array("pipe", "w")
                );

                $proc = proc_open($cmd, $desc, $pipes);

                $ret = stream_get_contents($pipes[1]);
                $err = stream_get_contents($pipes[2]);

                fclose($pipes[1]);
                fclose($pipes[2]);

                $retVal = proc_close($proc);

                if (func_num_args() == 2) $out = array($ret, $err);
                return $retVal;
        }

        //
        // INTERNAL TOOLKIT -- {STOP}
        //
}
?>
```

3. Pagina do Post – arduino.php

```
<html>
<head>
        <meta http-equiv="refresh" content="5;URL=http://190.168.120.120">
        <title> Sistema E-SOLAR </title>
        <meta name="description" content="Sistema Servidor autonomo com energia solar!!
">
        </head>
<?php

$port = fopen("/dev/ttyUSB0", "w");
if ($_POST['estado']=="30min")
{
   echo "Ativou 30 Minutos!";
   fwrite($port, "e"); //envia pela saida serial para o arduino
}
   if ($_POST['estado']=="1hora")
   {
      echo "Ativou 1 Hora!";
      fwrite($port, "f");
   }
      if ($_POST['estado']=="2horas")
      {
         echo "Ativou 2 Horas!";
         fwrite($port, "g");
      }
         if ($_POST['estado']=="3horas")
         {
            echo "Ativou 3 Horas!";
            fwrite($port, "j");
         }
                                    if ($_POST['estado']=="Desl")
                                    {
                                            echo "Desligando!";
                                            fwrite($port, "d");
                                    }
fclose($port);
?>
<body>
<p><h3>Enviando comando...</h3>
</body>
</html>
```