FATEC SANTO ANDRÉ
Tecnologia em Mecatrônica Industrial

Adriano Oliveira da Silva

Júlio César Batista Galera

Rafael Marcos Batista dos Santos

SUPERVISÃO E CONTROLE DE ENERGIA EM PONTO DE TOMADA

Santo André
2017

Adriano Oliveira da Silva

Júlio César Batista Galera

Rafael Marcos Batista dos Santos

SUPERVISÃO E CONTROLE DE ENERGIA EM PONTO DE TOMADA

Trabalho de conclusão de Curso apresentado a FATEC Santo André para obtenção do título de Tecnólogo em Mecatrônica Industrial
Orientador: Prof. Me. Pedro Adolfo Galani

Santo André
2017

LISTA DE PRESENÇA

Santo André, 21 de Dezembro de 2017

LISTA DE PRESENÇA REFERENTE À APRESENTAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO COM O TEMA:

"SUPERVISÃO E CONTROLE DE ENERGIA EM PONTO DE TOMADA"

DOS ALUNOS DO 6º SEMESTRE DESTA U.E.

BANCA

PRESIDENTE:
PROF. PEDRO ADOLFO GALANI _____

MEMBROS:
PROF. MURILO ZANINI DE CARVALHO _____

PROF. FERNANDO GARUP DALBO _____

ALUNO:
ADRIANO OLIVEIRA DA SILVA Adriano Oliveira da Silva

JULIO CESAR BATISTA GALERA Julio Cesar B. Galera

RAFAEL MARCOS BATISTA DOS SANTOS Rafael Marcos Batista dos Santos

FICA CATALOGRÁFICA

S586s

Silva, Adriano de Oliveira da

Supervisão e controle de energia em ponto de tomada / Adriano de Oliveira da Silva, Júlio Cesar Batista Galera, Rafael Marcos Batista dos Santos. - Santo André, 2017. – 66f: il.

Trabalho de Conclusão de Curso – FATEC Santo André.
Curso de Tecnologia em Mecatrônica Industrial, 2017.

Orientador: Prof. Pedro Adolfo Galani

1. Controlador. 2. Tomada inteligente. 3. Sensores. 4. Plataforma microcontrolada. 5. Dispositivo. I. Galera, Júlio Cesar Batista. II. Santos, Rafael Marcos Batista dos. III. Supervisão e controle de energia em ponto de tomada.

629.89

AGREDECIMENTOS

A Deus por ter nos dado saúde e força para superar as dificuldades.

A nossos familiares pelo incentivo, pela força e carinho.

Aos nossos colegas de classe que participaram das pesquisas de campo.

A esta faculdade, seu corpo docente, direção e administração que nos oportunizaram a janela do conhecimento.

Ao nosso orientador Pedro Adolfo Galani, pelo suporte no pouco tempo que tinha disponível, pelas suas correções e auxílios.

Bom, agradecemos todas as pessoas que fizeram parte dessa etapa decisiva das nossas vidas.

“Deixe o futuro dizer a verdade, e avaliar cada um
de acordo com seus trabalhos e suas conquistas.”

NIKOLA TESLA

RESUMO

Esse trabalho tem por objetivo a construção de um dispositivo que energizará ou desenergizará um equipamento que estará conectado a um ponto de tomada, com a obtenção dos dados deste mesmo ponto através de sensores de grandezas elétricas de consumo através de sensores apropriados. Posteriormente, esses dados são processados e transmitidos para uma plataforma microcontrolada com um transmissor *Wi-Fi*. A plataforma microcontrolada terá a função de transmitir os dados obtidos para a nuvem como meio de comunicação um roteador *Wi-Fi* e o meio de transmissão as nuvens, para que o usuário possa através de um supervisório (*smartphone*), monitorar a situação deste mesmo ponto em tempo real e em qualquer lugar do planeta, necessitando somente da conexão com a *internet* , podendo também alterar as condições para a energização deste ponto distante, com o mesmo processo para o meio de comunicação, mas desta vez com a unidade remota nas mãos do usuário, enviar a configuração pré-estabelecida pelo mesmo através da nuvens e o roteador e a plataforma microcontrolada funcionarem como receptores, e através de suas lógicas somadas aos dados enviados pelo usuário, atuarem diretamente sob o dispositivo, famoso com o nome de “tomada inteligente”. Adiciona-se assim ao usuário uma maior segurança sob seu estabelecimento e uma forma que possa conscientizá-lo e ter um controle sob o consumo de energia elétrica, que pode beneficiar o mesmo e o meio ambiente.

Palavras-chave: Controlador. Tomada Inteligente, Tele Medição, Consumo, Comando Remoto.

ABSTRACT

The objective of this work is the construction of a device that will energize or deenergize an equipment that will be connected to a point of outlet, obtaining the data of this same point through sensors of electrical quantities of consumption through appropriate sensors. Later, this data is processed and transmitted to a microcontrolled platform with a Wi-Fi transmitter. The microcontrolled platform will have the function of transmitting the obtained data to the cloud as a means of communication a Wi-Fi router and the means of transmission the clouds, so that the user can through a supervisor (smartphone), monitor the situation of this same point in real time and anywhere on the planet, requiring only the connection to the internet, and may also change the conditions for energizing this far point, with the same process for the communication medium, but this time with the remote unit in the hands of the user, send the configuration pre-established by the same through the clouds and the router and the microcontrolled platform function as receivers, and through their logic added to the data sent by the user, act directly under the device, famously called "smart socket." This gives the user greater security under his / her establishment and a way that can make him / her aware and have a control under the consumption of electric energy, which can benefit the same and the environment.

Keywords: Controller. Smart Plug, Tele Metering, Consumption, Remote Control.

LISTA DE IMAGENS

LISTA DE ABREVIATURAS E SIGLAS

IDE – *Integrated Development Environment* (Ambiente de Desenvolvimento Integrado).

IOT – *Internet Of Things* (Internet das Coisas).

OSI – *Open System Interconnection* (Sistema aberto de Interconexão).

UTR – Unidade de Terminal Remota.

TC - Transformador de corrente.

HTTP – *HyperText Transfer Protocol* (Protocolo de transferência de hipertexto).

HTTPS – *Hyper Text Transfer Protocol Secure* (Protocolo de transferência de hipertexto seguro).

GPL – *General Public License* (Licença publico geral).

LGPL – *Lesser General Public License* (Licença pública geral menor).

CAD – Computer Aided Design (Desenho assistido por computador).

MIT – *Massachusetts Institute of Technology*.

SCADA - *Supervisory Control and Data Acquisition* (Controle de Supervisão e Aquisição de Dados).

JSON – *JavaScript Object Notation.*

HTML – *HyperText Markup Language.*

SQL – *Structured Query Language.*

API – *Application Programming Interface (*Interface de programação de aplicações).

Wi-Fi – *Wireless Fidelity (*Fidelidade sem fio).

TCP – *Transmission Control Protocol* (Protocolo de Controle de Transmissão).

IP – *Internet Protocol* (Protocolo de internet).

RTC – *Real Time Clock* (Relógio em tempo real).

MAC - *Media Access Control* (Controle de acesso de mídia).

SUMÁRIO

1 INTRODUÇÃO

Devido ao grande avanço da tecnologia, a automação vem ganhando amplo espaço nas residências. Tratando-se de tarefas de controle manual, podemos observar a necessidade de praticidade e inteligência para um controle de maior eficiência, monitoramento e segurança que regem em nosso cotidiano.

Algumas destas atividades realizadas pelos humanos, que se tornam suscetíveis a erros, muitas vezes submetem a riscos prejudiciais, excesso de consumo de tempo e energia elétrica, que acabam se tornando fragilidades.

Pensando em uma solução viável para estas “fragilidades” citadas, a IOT (*Internet of things*) também conhecido como internet das coisas, é um sistema inteligente, que de grande partida visa desenvolver um sistema que possa nos auxiliar com nossas falhas ou até mesmo tornar-se nosso “assistente” a partir dos dispositivos controlados remotamente pelo usuário, ou um sistema autocontrolado com as características pensadas pelo desenvolvedor. Por meio desse sistema inteligente, será possível obter-se a interação com dispositivos eletrônicos utilizados no dia a dia que demandam de nossa presença para manipulá-los, como por exemplo, o controle de demanda, tensão e corrente, entre outros, nos mostrando que podemos ter um controle sob uma área de uma forma simples e prática, como por exemplo, a utilização de *smartphones* para o monitoramento e/ou acionamento de outros periféricos.

Com base nos conhecimentos obtidos ao longo do curso, este projeto tem como objetivo utilizar a internet das coisas com o desenvolvimento de um dispositivo que tem por sua função controlar um ponto de energia em ponto de tomada através de uma unidade remota como mestre (*smartphone),* e como terceiro monitorar o consumo de energia em tempo real deste mesmo ponto com um determinado equipamento energizado no mesmo, utilizando novamente o *smartphone,* mas desta vez como supervisório.

Com o emprego da IOT (*Internet of Things*), pretende-se desenvolver uma aplicação para supervisionar e controlar pontos de consumo de energia elétrica,

utilizando dados a serem armazenados nas nuvens, para obtermos um controle de um ponto de energia, tendo em vista três vertentes:

• Controlar a demanda do consumo de energia de modo que a carga seja desligada quando tender a um limite previamente estabelecido;

• Assegurar ao usuário que uma determinada carga não possa ser ligada ou que se mantenha desconectada;

• Facilitar as atividades cotidianas de modo que as cargas possam ser ligadas ou desligadas remotamente, ou ligadas e desligadas em horários pré-estabelecidos.

Visando atender essas vertentes, pretende-se utilizar unidades remotas microcontroladas, que farão as *telemedições* e *telecomandos,* conectadas a roteadores e servidores internet. Um eventual usuário, através de um aplicativo *open source* em seu celular, poderá interagir com o sistema.

Este trabalho está dividido da seguinte forma:

O primeiro capitulo, está presente a fundamentação teórica onde contém o embasamento teórico para a realização do projeto Supervisão e controle de energia em ponto de tomada .

No segundo capitulo, encontra-se a Metodologia utilizada para a elaboração do projeto.

O terceiro capitulo, encontra-se Desenvolvimento do projeto, que descreve o como foi o andamento para a realização do projeto.

2 FUNDAMENTAÇÃO TEÓRICA

Neste primeiro capítulo foram introduzidos os “pilares” para o desenvolvimento do projeto, desde a obtenção de dados de um ponto energizado, até a manipulação pelo usuário através de uma unidade remota.

2.1 Medição de tensão e corrente

O termo tensão refere-se a uma representação de quantidade de energia envolvida na movimentação de uma carga entre dois pontos de um sistema elétrico (Boylestad, 2004).

Pode-se definir tensão sendo a quantidade de energia gasta de 1 joule para deslocar a carga de 1 coulomb onde a diferença de potencial será de 1 volt, essa diferença de potencial é a responsável pelo fluxo de cargas de um circuito elétrico, quando percorrida por um condutor surge o termo também conhecido como corrente elétrica.

Para medir tensão, é comum a utilização de voltímetros, quando se deseja medir a diferença de potencial em um elemento de um circuito, deve se conectar o voltímetro em paralelo com o mesmo, tendo em vista que a resistência interna do voltímetro não é infinita, o voltímetro drena uma parte da corrente que passaria no circuito. Para que isso não ocorra, a resistência interna do voltímetro deve ser muito alta, desprezando esta corrente drenada.

Sua unidade é conhecida como Volt em homenagem a Alessandro Volta, e sua equação (2.1) é apresentada abaixo:

$$\boldsymbol{V = \frac{\Delta W}{\Delta Q}} \qquad \textbf{(2.1)}$$

Onde:

"V" é a Tensão Elétrica em volts [V].

"ΔW" é a quantidade de energia potencial em joules [J].

"ΔQ" é a quantidade de cargas em Coulombs [C].

A corrente elétrica é o fluxo ordenado de elétrons quando é aplicado ao condutor elétrico uma diferença de potencial onde o deslocamento deste fluxo tenta se equilibrar após a ação do campo elétrico gerado (Boylestad,2004).

A intensidade da corrente elétrica pode ser definida como a razão entre o módulo da quantidade de cargas que atravessa o condutor em um intervalo de tempo.

A unidade de medida de corrente elétrica é o Ampére (A), em homenagem a André Marie Amperé, e sua equação (1). é apresentada na equação 2.2 abaixo:

$$\boldsymbol{I = \frac{\Delta Q}{\Delta T}} \quad \textbf{(2.2)}$$

Onde:

"I" é a Corrente Elétrica em (Amperés) [A].

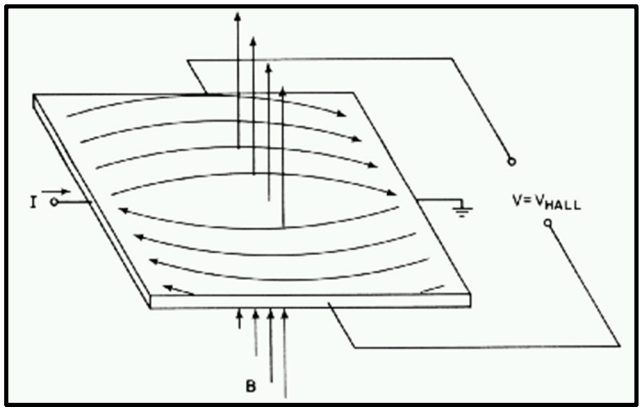
"ΔQ" é a quantidade carga elétrica na secção transversal em Coulombs [C].

"Δt" é a variação de tempo que observamos em segundos [s]

2.1.1 Sensor por efeito Hall

Também é possível empregar-se sensores de tensão por efeito Hall (Efeito descoberto por Edwin H. Hall no final do século XIX) os quais possuem a capacidade de medir tensão contínua e alternada em um único instrumento. Certos componentes são desenvolvidos especificamente para condicionar níveis de tensão (Belchior, 2014).

Figura 2.1: representação do efeito hall.



Fonte: http://www.eletrica.ufpr.br/edu/Sensores/2000/neis/, Acesso em: 05/2017

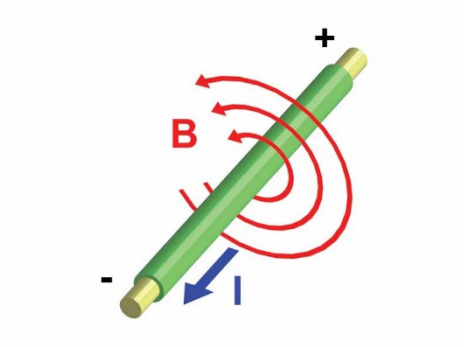
O funcionamento de sensores de efeito Hall consiste na geração de um campo elétrico transversal a um condutor, quando este está imerso em um campo magnético e é percorrido por uma corrente elétrica. A faixa de operação desse componente é de 10 a 500V. Para realizar a medida, é preciso energizá-lo com tensões de ±12V ou ±15V. Trata-se de um medidor com boa linearidade, ótima imunidade contra ruídos, possui uma grande largura de banda e ótima precisão. A tensão Hall é proporcional ao produto da corrente da tensão de entrada com o valor do campo magnético incidente perpendicularmente (Belchior, 2014).

2.1.2 Sensor SCT-013

O sensor de corrente não invasivo (SCT-013), tem a finalidade de extrair informações relacionadas ao consumo de energia.

Seu funcionamento é semelhante ao de um alicate amperímetro, onde obtém-se parâmetros através do campo magnético gerado por um eletrocondutor no qual está sendo envolvido pelo sensor.

Figura 2.2: campo magnético junto a um condutor onde circula corrente elétrica.



Fonte:http://www.wikipremed.com/image.php?img=010403_68zzzz141800_Electromagnetism_68.jpg&image_id=141800, Acesso em: 05/2017

Conforme informações retiradas do YHDC *Split core current transformer*, pode-se obter as seguintes características:

2.2 Medição da Potência

A potência é uma grandeza que mede o trabalho (conversão de uma forma de energia em outra) realizado por um dispositivo em um determinado período de tempo, onde pode ser obtida pela relação de tensão e corrente. A Equação 2.3(Robert. L. Boylestad, 2004) demostra como calcular a partir de tensão e corrente, a potência consumida de um dispositivo.

$$\boldsymbol{P = V \times I} \qquad (\mathbf{2.3})$$

Por meio da lei de Ohm, pode-se obter a potência dissipada pelo sistema ou dispositivo:

$$\boldsymbol{P = V \times I \rightarrow P = V \times \left(\frac{V}{R}\right)}$$

$$\boldsymbol{P = \frac{V^2}{R}} \qquad (\mathbf{2.4})$$

OU

$$\boldsymbol{P = V \times I \rightarrow P = (I \times R) \times I}$$

$$\boldsymbol{P = I^2 \times R} \qquad (\mathbf{2.5})$$

Onde:

- P é a potência medida em Watts (W);
- V é a tensão medida em Volts (V);
- I é a corrente medida em Ampere (A);
- R é a resistência medida em Ohm (Ω).

2.2.1 Potência Aparente

Segundo Ademaro A. M. B. Cotrim (2009), a potência aparente é a potência total que o dispositivo consome da rede de alimentação, é a soma da potência ativa e reativa, onde a unidade de medida é volt-ampere (VA). A equação 2.6 é utilizada para obtenção da potência aparente.

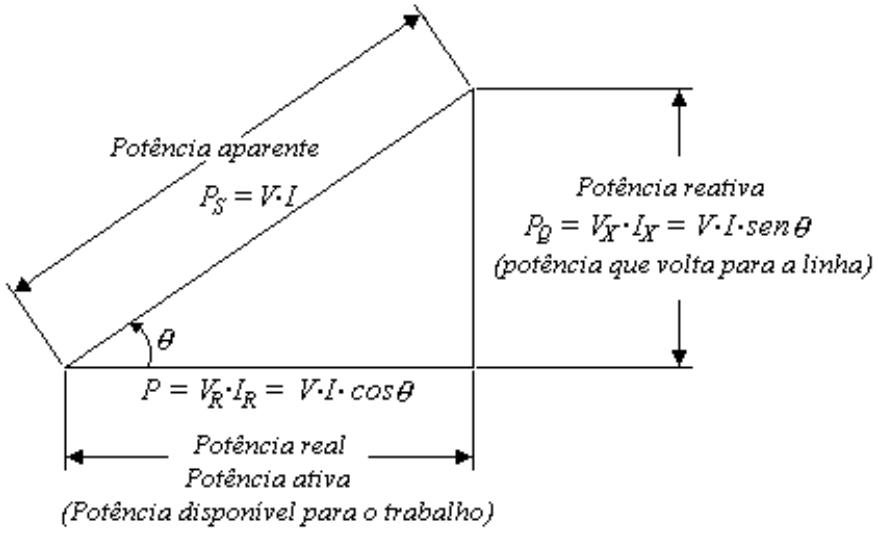
$$\boldsymbol{S = \sqrt{P^2 + Q^2}} \qquad (\mathbf{2.6})$$

Onde:

- S é a potência aparente (VA)
- P é a potência ativa (W)
- Q é a potência reativa (VAR)

Pode ser representada a relação entre a potencia ativa (potência que realmente realiza trabalho), reativa (potência que volta para a linha) e aparente pelo um triângulo de potência, mostrado a seguir:

Figura 2.3: triângulo de Potência.



Fonte: http://www.ebah.com.br/content/ABAAAAQDoAC/eletricidade?part=19,

Acesso em: 2017

2.2.2 Demanda

A demanda de energia, seja pela necessidade de uma pessoa ou de uma empresa de fazer o acionamento de um dispositivo ou maquina elétrica, é a soma das cargas instaladas atuando no mesmo intervalo de tempo, medido em quilowatts (kW), ou seja, é a capacidade máxima do sistema elétrico em determinado momento (Ademaro A. M. B. Cotrim, 2009).

Os consumidores diversas vezes trocam os significados de demanda e consumo. O consumo é medido em quilowatts x hora (kwh), é relação do período de tempo em que uma carga é alimentada, difere-se assim da demanda.

Segundo Ademaro A. M. B. Cotrim (2009), a escolha dos fatores de demanda em uma instalação industrial ou residencial, devem levar em consideração alguns aspectos, tais como:

- Qual atividade será realizada na devida instalação;
- Equipamentos de aquecimento e refrigeração, devem ser considerados os fatores climáticos;
- Flexibilidade caso necessite fazer alterações em equipamentos.

A utilização de fatores de demanda muito baixa leva o circuito ao subdimensionamento. Nesse caso, haverá problema no horário de pico de consumo de energia, pois no momento de pico vamos ter uma corrente real maior do que a corrente dimensionada.

2.2.3 Controle de demanda

São componentes eletrônicos cuja função é manter valores de demanda dentro de limites pré-determinados, visando melhor eficiência não só para consumidores reduzir custos com a energia elétrica, como também, para a

concessionária que pode oferecer um sistema elétrico com melhor qualidade, partindo do momento que funcionará de forma dimensionada evitando interrupções (Fernando Silva Ozur et al., 2011).

Existem dois tipos de controladores de demanda:

- Convencionais;
- Inteligentes;

Um controlador de demanda convencional opera de maneira sem interrupções dentro de um intervalo de integração, pois utiliza medição por média móvel e controle por níveis ou por controle de projeção simples. Um controlador de demanda inteligente funciona de maneira eficiente, permitindo que a demanda cai de forma natural, adiando seu controle (Fernando Silva Ozur et al., 2011).

O método de controle de demanda define a forma como será monitorado e controlado, existem alguns métodos de controle, tendo o algoritmo de janela móvel. o método de retas de cargas ou retas inclinadas como principais (Fernando Silva Ozur et al., 2011).

2.2.4 MDGE (MEDIDORES DE GRANDEZA ELÉTRICA)

É um dispositivo eletrônico cuja finalidade é medir valores de potência, tensão e corrente consumida em um sistema elétrico de forma digital. Medidores de grandeza elétrica podem ser eletromecânico ou digital, porém o mais comum de ser utilizado para fazer medição em sistema elétrico são os digitais. (Guadagnin et al, 2011).

Os medidores de energia elétrica, são utilizados pelas concessionárias fornecedoras de energia para se obter os valores de consumo de cada usuário e a partir das variáveis obtidas, estabelecer uma tarifa “justa” para cada consumidor.

Nas industrias, facilmente podemos ver cargas que consumem dois tipos de potencias: potência reativa é usada para criar ou manter o campo eletromagnético das cargas indutivas e a potência ativa que realiza o trabalho esperado. Portanto, é

necessária uma grandeza que relacione tanto a potência reativa, quanto a potência ativa. O fator de potência é a grandeza elétrica utilizada para analisar o consumo de potência nesse caso. (Guadagnin et al, 2011).

O medidor obtém os sinais elétricos a partir das grandezas como: tensão, corrente, potência ativa, potência reativa, fator de potência, entre outros. E disponibiliza em seu mostrador digital os valores lidos (Guadagnin et al, 2011).

2.3 Internet das coisas (*Internet of things – IOT*)

Internet das coisas denomina-se da conectividade de dispositivos e sistemas empregados com sensores e atuadores a operar em conjunto com algum outro periférico, seja físico ou virtual, que de alguma forma possa proporcionar uma melhor qualidade de vida, seja em produtividade, lazer, praticidade e outros fatores que uma pessoa passe a ter uma "vida conectada". Os aspectos que a IOT (*Internet Of Things)* trazem, podem também proporcionar saúde, segurança, eficiência entre outras intenções de seu uso (Evans, 2011).

Vem de um segmento da evolução humana, onde substituímos todos os processos com ações manufaturadas por processos automatizados e controlados, diminuindo cada vez mais o esforço humano e tendo um resultado mais rápido e eficiente de qualquer invenção/transformação. A expressão em tornar nossas vidas conectadas é fazer com que os componentes integrados a ela sejam controlados pela ponta de nossos dedos em qualquer hora em qualquer lugar, o que se assemelha a internet que vivenciamos ao enviar uma mensagem, realizar registros ou transferências de dados, mas agora com o aspecto físico.

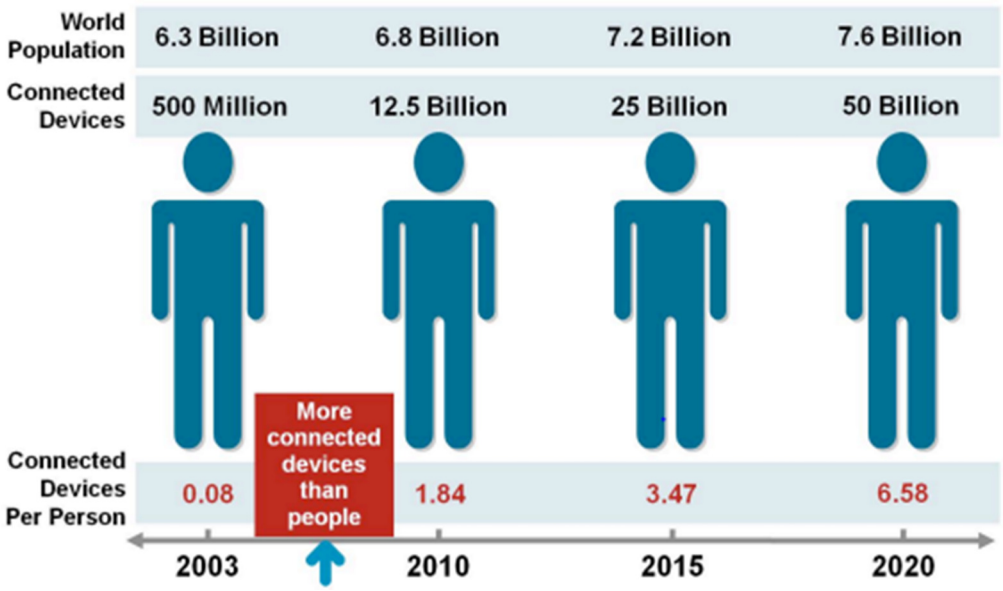
Conforme Evans (2011), "Ações das coisas isento de um meio dinâmico operado por uma tecnologia invisível (*internet*) ".

Este é um próximo passo para a internet do futuro, e tem vindo conectando cada vez mais a vida dos usuários. Alguns *web services* tem contribuído para tal evolução, fornecendo funções que interligam determinadas aplicações com o usuário e com o sistema, através de protocolos telecomandados (conhecido também com banco de dados).

Como esta tecnologia é tendência do futuro, nota-se a necessidade de ser cada vez mais acessível para todos, no entanto, adaptar o aplicativo de uma forma que tenha a aparência da tecnologia que é presenciada, que nos dias atuais são os *smartphones.* A chegada dos *smartphones* foi essencial para o avanço da *IOT,* pois abrangeu uma grande população a conectar-se com dispositivos. Uma pesquisa sobre internet das coisas, realizou um levantamento de dados e que em 2003, por exemplo, com 500 milhões de dispositivos conectados, (menos que 0.08 dispositivo por pessoa por todo o mundo) passou para 12.5 bilhões em 2010, (1,8 dispositivo por pessoa) com a chegada do primeiro *IPhone* desenvolvido por Steve Jobs em Janeiro de 2007 e a explosão de outros eletrônicos.

A internet das coisas nasceu por volta de 2008 e 2009, e que teremos 50 bolhões de dispositivos conectados em 2020 (Evans, 2011).

Figura 2.4: Utilização de dispositivos conectados ao longo dos anos.



Fonte: DAVE EVANS, The Internet of Things How the Next Evolution of the Internet Is Changing Everything, 2011, Acesso em 05/2017.

Sua metodologia mais utilizada consiste em enviar e receber dados com a nuvem através de uma unidade remota (mais comumente os *smartphones).* Os dados recebidos indicam instantaneamente a posição do que se deseja interagir de acordo com suas necessidades. Já os dados transmitidos são as variáveis que o usuário solicita ao sistema para uma determinada função, tendo como exemplo

automatização residencial têm-se vários periféricos a serem integrados a esta tecnologia, como controle de temperatura, luminosidade, segurança, dados de consumo entre outros (Evans, 2011).

Figura 2.5: Residência automatizada com utilização da IOT

Fonte: DAVE EVANS, The Internet of Things How the Next Evolution of the Internet Is Changing Everything, 2011.

2.3.1 Thingspeak

Segundo Maureira et al (Acesso em: 22/05/2017), *thingspeak* é uma plataforma de Internet de Coisas (IOT) que permite coletar e armazenar dados de sensores na nuvem e desenvolver aplicações IOT e que também fornece recursos que permitem analisar e visualizar dados através de geração de gráficos e atuar sobre os mesmos. Os dados são armazenados nos chamados canais, que fornece ao usuário uma lista de recursos.

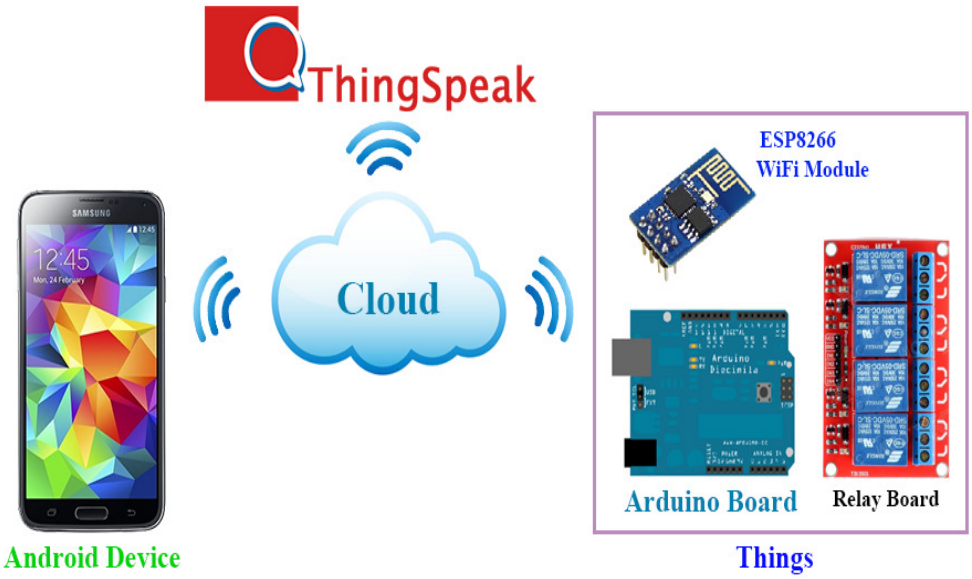
Cada canal permite que você armazene até 8 campos de dados, usando até 255 caracteres alfanuméricos cada. Existem também 4 campos dedicados para dados posicionais, que consistem em: descrição, Latitude, Longitude e Elevação (Maureira et al, Acesso em: 22/05/2017). Todos os dados recebidos são marcados com data e hora e recebem uma ID sequencial. Uma vez que um canal foi criado, os

dados podem ser publicados acessando a API ThingSpeak com uma 'chave de escrita', uma sequência alfanumérica única criada aleatoriamente para autenticação (Maureira et al, Acesso em: 22/05/2017.).

O *upload* de dados se dá através de requisições HTTP/HTTPS contendo os dados, canais onde devem ser escritos e uma chave de autenticação de escrita do canal, caso este não seja público (Lemos, 2016). Com essa plataforma pode-se enviar e receber dados gerados por um *Arduino* ou qualquer outro dispositivo com recurso para comunicação em rede. Esse serviço permite, por exemplo, usar o *Arduino* para gerar um *tweet*, criar uma rede social de dispositivos ou controlar a distância motores e outros dispositivos (Maureira et al, Acesso em: 22/05/2017.).

De acordo com Maureira, a API funciona conforme observado na Figura 2.6. Os dados são então carregados para a nuvem e, a partir daí, podem ser usados para vários fins. Por sua vez, os dados (como comandos ou a escolha de determinadas opções) podem ser reunidos e comunicados à nuvem, o que, por sua vez, envia essas mensagens ao objeto.

Figura 2.6: Ilustração de um escopo com o servidor *thingspkeak*.



Fonte: http://androidcontrol.blogspot.com.br/2015/06/android-iot-control-thingspeak.html, Acesso em 05/2017.

2.3.2 Open source

Open source ou código aberto refere-se a *softwares* sem direitos autorais *ou copyright* que foram desenvolvidos por alguém que permitiu o acesso da estrutura de seu trabalho, para que os usuários possam explorá-lo e modificá-lo como desejar, sem restrições. Mas vale ressaltar que *softwares* de código aberto são aqueles com licenciamento aprovado pela *Open Source Iniciative (OSI),* obedecendo dez requisitos definidos pela *The Open Definition,* diferentemente do código livre que obedece *The Free Software Definition,* e garantindo quatro requisitos de liberdade: execução, estudo, redistribuição e melhoria. Os softwares de código aberto são beneficiados por cada modificação realizada por uma comunidade, evoluindo com as inúmeras possibilidades de configurações que podem ser exploradas pelos usuários, mas sempre obedecendo à licença de distribuição ditada pelo desenvolvedor (Carvalho, 2013).

Para determinar o tipo de licença, que varia de acordo com o quão flexível podemos alterar o produto, devemos conhecer o modelo que será adotado, pois variam em características de licenciamento. Há licenciamentos que permitem a criação de um código proprietário a partir de um código já existente, assim como os que permitem utilização de seu código pagando por sua licença de uso (Carvalho, 2013).

Neste projeto tem-se como exemplo de código aberto o microcontrolador com uma configuração de compatibilidade para seu código disponibilizado pela plataforma *Arduíno*, liberado sob a licença GPL (*General Public License*), as bibliotecas microcontroladoras C/C++ sob LGPL (*Lesser General Public License*), e os esquemas e arquivos CAD sob *Creative Commons Attribution Share-Alike*. Uma plataforma que permite um universo de possibilidades para criação de projetos, podendo ser configurado de acordo com sua necessidade (Carvalho, 2013).

2.3.3 Roteadores

Os roteadores são aparelhos usados em uma rede de computadores com a finalidade de fazer o encaminhamento de pacotes dados, proporcionando conectividade de dispositivos como *smartphones*, celulares, *tablets*, etc; (Zerfos et al, Acesso em 05/2017).

Conforme (Kamienski et AL, 2014), o termo roteamento pode ser designados para dois processos distintos que acontece nos roteadores, a primeira busca por melhores caminhos (rotas) para enviar e receber dados, priorizando não só as transmissões mais curtas, como também as menos congestionadas. A segunda é o processo de enviar os pacotes de dados a um determinado aparelho ou para outro roteador existente, de acordo com as informações sobre a rota. Esse último processo é chamado como encaminhamento de dados.

O melhor caminho pode ser estabelecido através de modo estático ou dinâmico. O roteamento estático é construído manualmente pelo administrador do sistema, ou seja, uma pessoa será responsável por manter as informações de configuração atualizada. As informações a respeito da rota de dados são mantidas pelos roteadores por meio de tabelas de roteamento, onde cada linha basicamente identifica uma rede de destino final.

Roteamento dinâmico é mais eficiente que o estático, uma vez que pode resolver problemas complexos de roteamento mais rápido, por meio de um protocolo desenvolvido para alterar a rota dinamicamente dependendo da condição da rede.

2.4 UTR (Unidade de terminal remota)

Segundo Heng, 1995, UTR (unidade terminal remota) pode ser definida como um dispositivo de estação remota de um sistema de supervisão, desenvolvido para automatizar ambientes de supervisão e controle de uma planta de telecomando e telemedição.

Uma vez que a UTR interage com os equipamentos do mundo real, consideram-se os olhos, as orelhas e as mãos da estação principal. A inclusão de

um microprocessador na UTR permitiu descarregar o canal de comunicação, bem como o computador da estação principal executando algumas das etapas de processamento adicionais anteriormente realizadas pelo computador principal.

2.4.1 ESP 8266

A plataforma de conectividade permite alta integração de circuitos com a capacidade imensa de comunicação WiFi com outros dispositivos com praticidade e baixo custo.

Segundo KOLBAN (2016), o módulo ESP é uma plataforma completa, ou seja, pode ser utilizada para receber e enviar dados da rede WiFi de uma determinada aplicação, podendo ser uma plataforma independente possuindo seu próprio processador, ou sendo como *shield* para outros tipos de micro controladores.

Como se trata de um dispositivo desenvolvido recentemente, infelizmente possuímos pouco suporte no que diz respeito ao seu funcionamento, sua arquitetura e exemplos de utilizações.

Este dispositivo opera com o TCP/IP (conjunto de protocolos que permitem a comunicação de um dispositivo com a rede). Também comparado com o modelo OSI, onde cada camada possuí uma determinada tarefa. O modelo OSI em conjunto com TCP/IP formam a pilha de protocolos mista utilizado na internet.

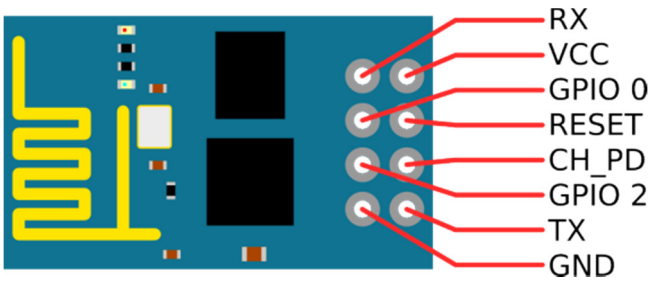
Conforme as informações retiradas do *Espressif, version 5.4*, o módulo ESP opera em três modos:

- *Deepsleep*: Somente o RTC é ligado e o resto do chip é desligado. A recuperação de memória do RTC pode salvar informações básicas de conexão WiFi.
- *Sleep*: Apenas o RTC opera, porém poderá ser “despertado” ao receber um comando para tal evento (MAC, host, temporizador RTC, interrupções externas).

- *Wakeup*: Neste estado todo o chip está em modo de operação.

A imagem e a tabela abaixo demonstram a relação de pinos do ESP e a função de cada terminal:

Figura 2.7: Ilustração da relação de pinos do ESP-01.

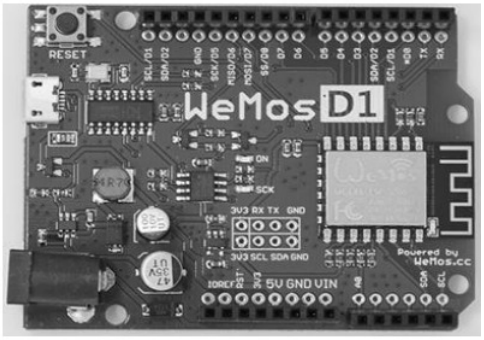


Fonte: https://blog.butecopensource.org/tag/esp8266/, Acesso em 05/2017

2.4.2 Plataforma Wemos

É um microcontrolador com módulo ESP8266 embutido (citado acima), que tem o mesmo princípio de funcionamento de outros microcontroladores. Ele é bastante semelhante ao *arduíno* UNO, onde pode-se utilizar a mesma linguagem de programação (IDE), porém com a diferença de algumas relações de pinos.

Figura 2.8: Ilustração do *Wemos* D1 R2



Fonte: http://www.instructables.com/id/Programming-the-WeMos-Using-Arduino-SoftwareIDE/, Acesso em 05/2017

Dadas essas diferenças de pinos, o *Wemos* se caracteriza por haver apenas uma entrada analógica, que pode vim a ser uma limitação para projetos que necessitam de duas ou mais entradas analógicas, mas se tratando de soluções rápidas e econômico é um dispositivo de grande exemplo e aplicabilidade, encontrado facilmente em lojas online ou comerciais de fins eletrônicos.

Figura 2.9: Comparação entre a plataforma *arduíno UNO* e *Wemos* D1 R2

Características\Plataforma	Arduíno	Wemos
Biblioteca familiar	✓	✓
Módulo *wi-fi* incluso	X	✓
Menor custo	X	✓
Necessita atualizações	✓	X
Mais de uma IO analógica	✓	X

Fonte: Autoria Própria, 2017

O *Wemos* terá a função a partir de uma lógica programável, controlar um *hardware* que no caso, será um sensor não evasivo e um relé, que efetuara o controle do consumo de corrente elétrica e o acionamento e desativação de uma determinada carga sucessivamente, obter os valores medidos pelo sensor de energia elétrica e enviá-los para uma remota através das nuvens.

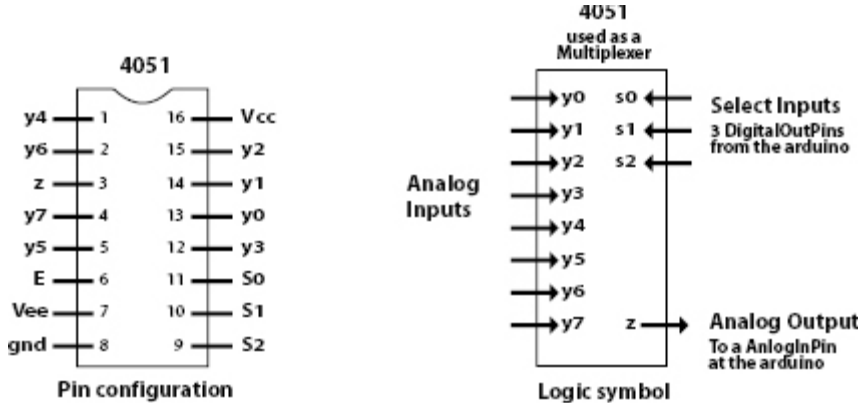
Porém como apresentado acima a plataforma *wemos* tem a desvantagem de possuir apenas uma IO analógica, que para o projeto que se deseja desenvolver é uma limitação, já que deseja-se obter dois sinais analógicos para medição de tensão e corrente. Pensando nisso, pode-se solucionar esta limitação com a multiplexação da única porta analógica existente na plataforma, para que haja mais de uma leitura, conforme destacado no próximo tópico.

2.4.3 C.I. CD4051 (Multiplexador)

Conforme informações retiradas do *Texas Instruments* CD4051B, CD4052B, CD4053B, o CD 4051 é um circuito integrado multiplexador/ demultiplexador de 8 canais, ou seja, pode-se escolher entre 8 entradas/saídas diferentes e selecionar uma por vez que deseja ler/escrever, onde é possível chavear tanto sinais analógicos como digitais.

Para selecionar o Pin que se deseja ler ou escrever, temos que usar três pinos digitais de seleção (S0, S1 e S2), mostrado na figura 2.10:

Figura 2.10: configuração dos pinos do CD4051



Fonte: https://playground.arduino.cc/Learning/4051, 2017

Não é possível ler ou escrever mais de um pino no CD4051 ao mesmo tempo, porque você só pode selecionar um pino por vez. Mas você pode ler e escrever nos pinos com bastante rapidez, ou seja, não ocorre atraso.

2.5 MIT APP INVENTOR

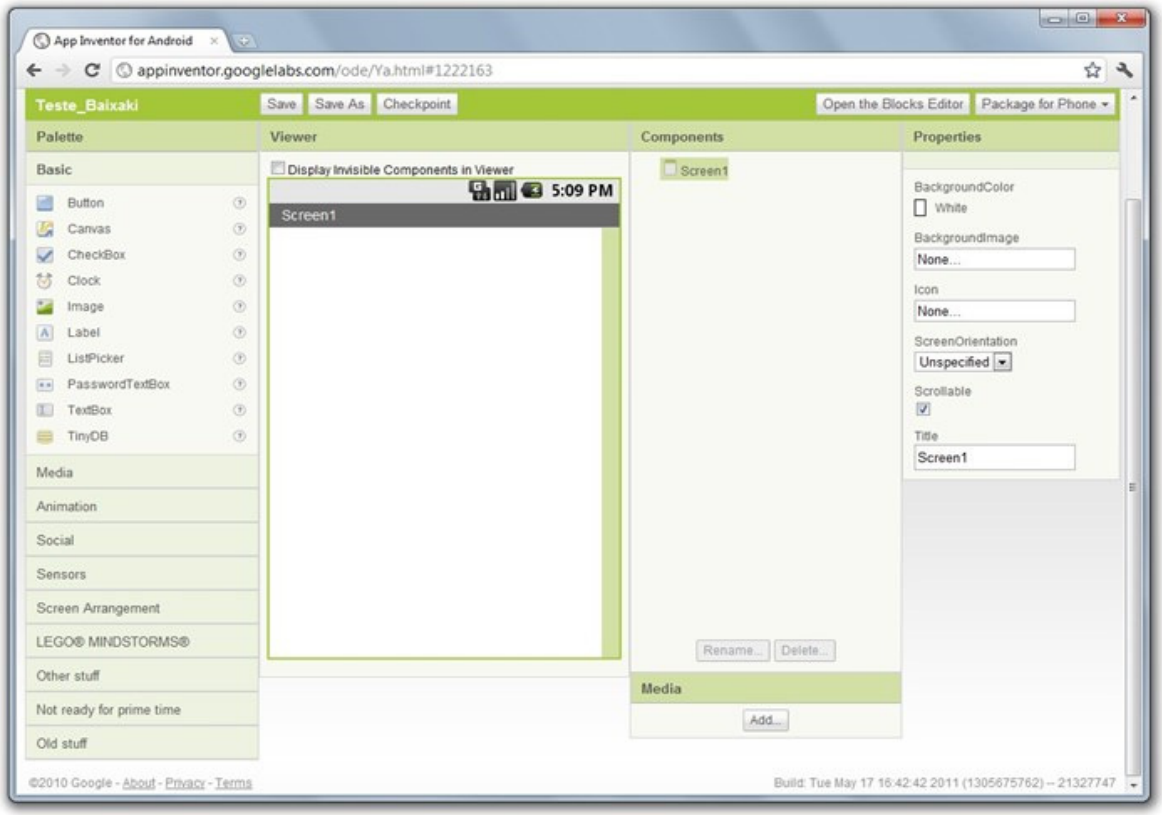
De acordo Tancicleide et al (2013), o *mit app inventor* é um ambiente de programação visual utilizado para criação de aplicativos para dispositivos moveis Android, baseado em blocos de códigos, sendo que seu diferencial é a possibilidade aos usuários criar aplicações de serviços baseados na web, leitura de códigos de barra, interação com sensores de orientação e geolocalização, etc.

Segundo Clark (2013), a empresa Google iniciou o desenvolvimento do App Inventor em 2009, entretanto dois anos depois, anunciou que não continuaria à ferramenta e o Centro de Mobile Learning do MIT (Massachusetts Institute of Technology) foi escolhido para sediar um servidor público para o App Inventor, além de tornar aberto o seu código-fonte.

A criação de aplicativos com o App Inventor é bastante intuitiva e não necessita de conhecimento prévio avançado em programação, onde os aplicativos criados podem ser utilizados em quaisquer dispositivos com a plataforma Android.

O desenvolvimento de uma aplicação nesta ferramenta é realizado através de duas janelas: *App Inventor Designer* e *Blocks Editor*. A janela *App Inventor Designer* é executada a partir do navegador e permite criar visualmente a interface do usuário, ao clicar e arrastar os componentes da *Palette*, tais como botões, caixas de texto, figuras, animações, sons, entre outros (Tancicleide et al, 2013), para o *Viewer* de acordo com a imagem a seguir.

Figura 2.11: Tela inicial do MIT App Inventory



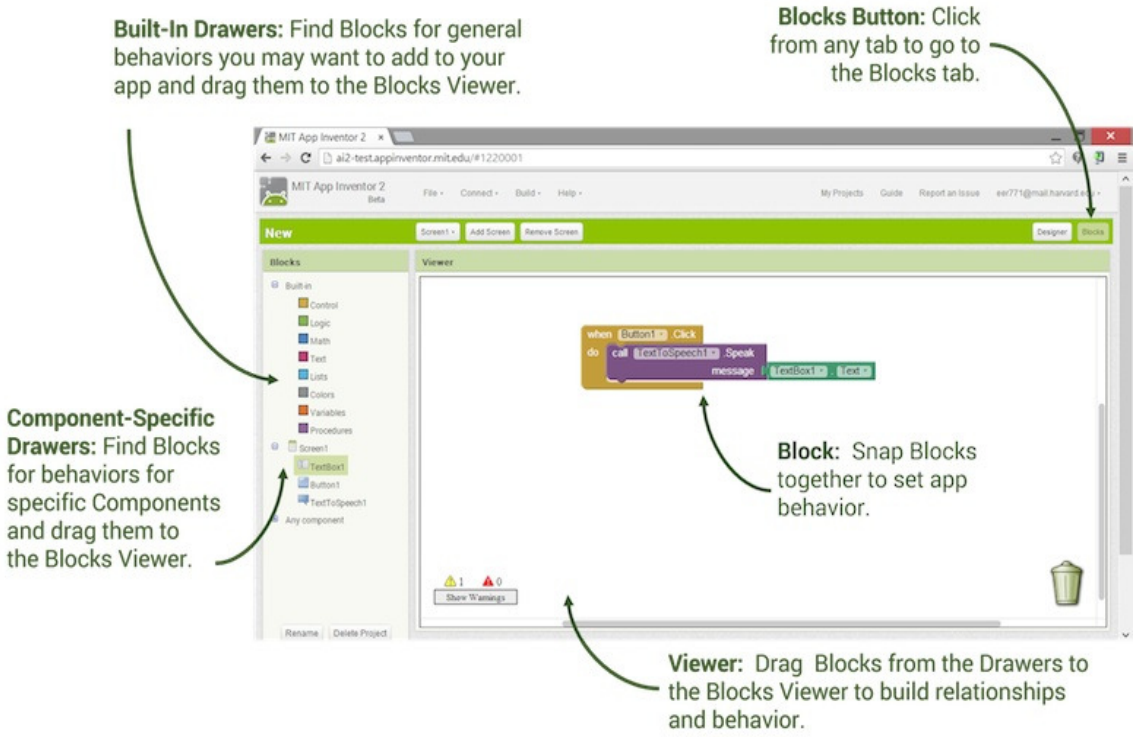
Fonte: appinventor.googlelabs.com/ode/ya.html#1222163, 2017

- *Pallete:* Escolha de componentes
- *Viewer:* visualização da organização dos componentes
- *Components:* lista de componentes utilizados

Properties: onde ocorre a definição das propriedades dos componentes

A janela *Blocks Editor*, permite controlar o comportamento dos componentes definidos na *App Inventor Designer*. Neste ambiente, são associadas ações para cada componente do aplicativo.

Figura 2.12: Ilustração do editor em blocos do MIT App Inventory



Fonte: http://appinventor.mit.edu/explore/designer-blocks.html, 2017

- *Blocks:* estão sempre disponíveis. Eles lidam com coisas como matemática, texto, logica e controles.
- *Component-Specific drawers:* blocos de componentes, correspondem aos componentes escolhidos no aplicativo
- *Viewer:* área onde são programados os blocos do aplicativo

3 METODOLOGIA

Neste capítulo foi relatado o caminho percorrido para construção lógica do trabalho intitulado Supervisão e Controle de Energia em Ponto de Tomada. Trata-se de criação de um dispositivo que é desenvolvido e testado nas dependências da FATEC Santo André.

3.1 Tema-problema e justificativa

O objetivo do trabalho foi desenvolver um dispositivo, que possa coletar informações de parâmetros de tensão, corrente e potência de determinado aparelho, e armazenar nas nuvens, a fim de proporcionar ao usuário poder de monitoramento e controle de um ponto especifico de energia. A principal justificativa é a necessidade de praticidade e inteligência para um controle de maior eficiência, monitoramento e segurança no uso dos recursos elétricos nas residências.

3.2 Etapa teórica e física na elaboração do trabalho

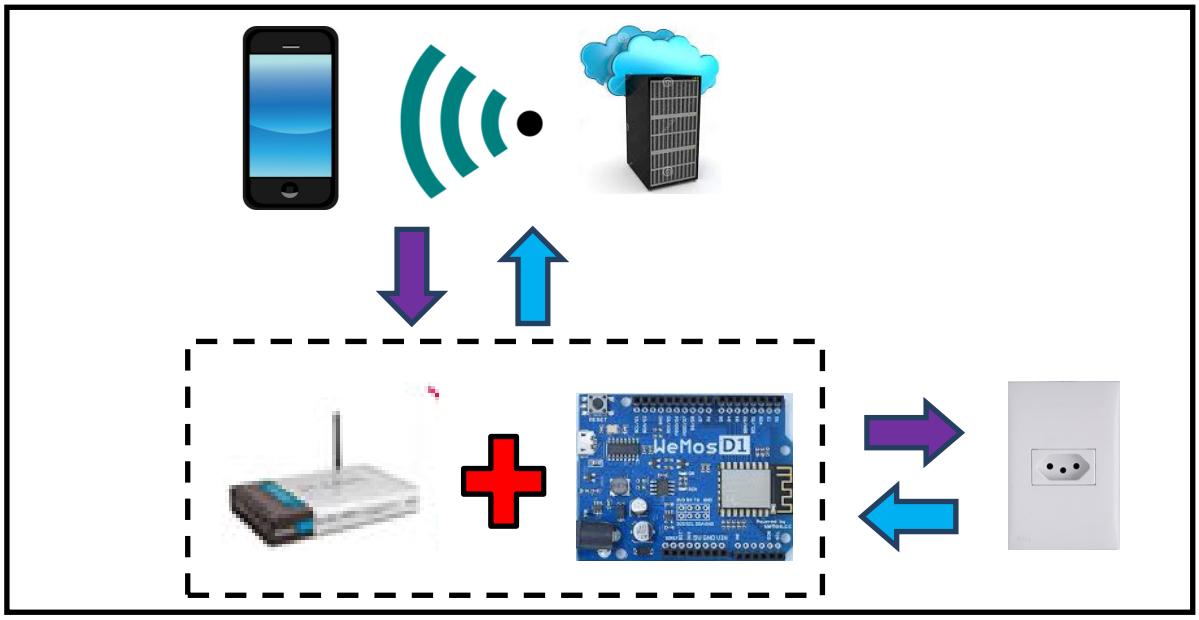
Após definir o tema-problema e sua justificativa partiu-se para:

Primeira etapa: reunião com os integrantes do grupo e o professor orientador para traçar os temos das pesquisas teóricas que dão sustentação ao desenvolvimento do projeto. Marcamos um dia por semana para reunião onde o grupo apresenta as tarefas determinadas para análise pelo orientador.

Segunda etapa: o levantamento bibliográfico das teorias em livros e artigos de autores renomados, bem como dissertações e teses deu-se nas bibliotecas virtuais em sítios fidedignos e em pdf direcionados ao tema. Após a leitura dos materiais que dão embasamento teórico do projeto constitui-se o capitulo 2 - Fundamentação Teórica.

Terceira etapa: construção do escopo do projeto para melhor visualização, conforme figura 3.1.

Figura 3.1: Escopo do projeto



Fonte: autoria própria, 2017

Quarta etapa: relação dos materiais para construção do projeto. Consultas em diversas lojas físicas e sites para obter o menor custo. Em seguida, a obtenção dos componentes, conforme tabela 3.2.

Tabela 3.2: Materiais utilizados e Orçamento

Operacional	Orçamento
Microcontrolador (Wemos)	R$ 25,80
Módulo Sensor de Tensão	R$ 16,20
Módulo Sensor de Corrente	R$ 27,00
Multiplexador (CD 4051)	R$ 2,20
Caixa com Tomada	R$ 25,00
Diversos componentes eletrônicos (placa, resistor, transistor, cabos, etc;)	R$ 42,50
Total:	R$ 138,70

Fonte: autoria própria, 2017

Quinta etapa: programação do *Wemos* (microcontrolador). Desenvolvimento do programa do módulo mestre no *software* arduino para fazer comunicação com a internet.

Sexta etapa: testes iniciais nos sensores de tensão e corrente.

Sétima etapa: criação do servidor web pelo *pythonanywhere*.

Oitava etapa: teste de comunicação entre o microcontrolador (*wemos*) e o servidor *web*.

Nona etapa: montagem do dispositivo que compreende: junções dos sensores de corrente e de tensão, transferência da programação desenvolvida para o *wemos*, e testes finais.
Terminada as etapas, segue-se para o desenvolvimento do projeto.

4 DESENVOLVIMENTO DO PROJETO

Este capitulo foi dedicado ao desenvolvimento do dispositivo que foi capaz de coletar informações de tensão, corrente e potência de um ponto especifico de energia, a fim de proporcionar ao usuário poder de monitoramento e controle, em um sistema supervisório.

4.1 Testes iniciais dos sensores

Para construção do dispositivo foi necessário um relé para fazer o controle da carga, sensores de corrente e tensão para captura de valores, para serem processados pelo microcontrolador. Foram realizados testes para determinar se os resultados obtidos na leitura e controle seriam confiáveis para que não apresentassem problemas, comprometendo a confiabilidade do projeto.

Sensor de tensão: A ideia inicial para medir a tensão na carga era a utilização de um transformador rebaixador com uma ponte retificadora, na qual seria medida a variação do secundário na entrada analógica do microcontrolador em relação ao primário onde estaria a carga.

Este módulo é constituído por resistores para ajuste, um capacitor para filtro e um optoacoplador:

Figura 4.1: Módulo sensor de tensão

Fonte: www.confrariadaescala.com.br/modulo-sensor-de-tensao-ac-p8-002014-gbk-robotics.html, 2017

A função do optoacoplador nesta aplicação é detectar a presença de tensão e saturar proporcionalmente a tensão de leitura no seu emissor.

Sensor de corrente: foram testados dois tipos de sensores de corrente, para que nosso dispositivo pudesse ser mais preciso e confiável possível.

O sensor não invasivo YHTC modelo SCT-013 100A foi acoplado entorno do cabo, onde obtém o valor através do campo eletromagnético gerado por um eletrocondutor.

A partir dos testes realizados com o sensor SCT-013, verificou-se que o mesmo não seria uma aplicação prática para o projeto, pois seus sinais de resposta eram inviáveis para calibração no microcontrolador como também para a instalação no protótipo. Visto isso, partiu-se para a utilização de um sensor mais familiarizado que teve sido apresentado para o grupo no desenvolvimento do projeto, o sensor ACS714.

O sensor ACS714 -30A a +30A possui um tamanho mais adequado para aplicação em tomada de ponto específico na qual transforma a variação de corrente em variação de tensão pelo efeito hall. A programação no microcontrolador como sua calibração para a sensibilidade que apresenta o tornou aplicável neste projeto pois atende as escalas de medições adequadas de corrente, podendo operar de -30A a +30A em tensão alternada ou contínua.

Figura 4.2: Sensor de corrente ACS714 -30A a +30A

Fonte: https://eletronicos.mercadolivre.com.br/pecas-e-componentes-eletricos-sensores/sensor-de-corrente-acs714, 2017

4.2 Programação do microcontrolador

A programação do dispositivo foi feita através do *software* do arduíno, onde pode ser encontrado no site do fabricante do controlador.

A programação no microcontrolador é feita em linguagem C, no Apêndice A, estão as principais programações feitas em nosso dispositivo.

4.3 Comunicação sem fio e funcionalidade

O modulo WIFI esp8266 integrado ao microcontrolador wemos proporciona conectividade junto ao roteador para enviar os dados coletados, que transmitirá ao um servidor que fará o envio e recebimento de dados a um sistema de supervisão e controle em diferentes plataformas como: *Android, IOS* ou *Scada*.

A placa *wemos* D1 R2 foi usada como mestre e escravo devido sua facilidade de desenvolvimento e por possuir um modulo WIFI esp8266 acoplado, porém é limitado por possuir uma porta analógica, para solucionar esse problema optamos por utilizar dois microcontroladores *wemos* por proporcionar um dispositivo mais barato e leitura confiáveis de valores.

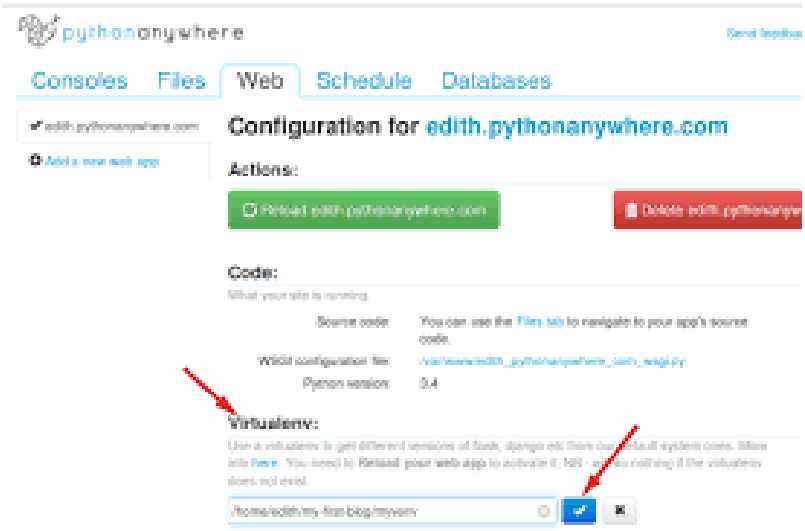
O roteador utilizado no projeto é o mesmo que tem nas residências, e através da configuração do IP do roteador no servidor, o usuário no sistema de supervisão e controle acessa as informações coletadas em *tablets* e *smartphones*, conforme ilustra a figura 4.3.

Figura 4.3: Roteador

Fonte: http://www.buscape.com.br/roteador, 2017

Optamos por utilizar o *pythonanywhere* como servidor, por ser gratuito para aplicações pequenas que não possuem muitos visitantes, então será suficiente para nossa aplicação. O *pythonanywhere* é uma plataforma versátil, robusta, segura, onde não se encontra nenhum tipo de burocracia para utiliza-la, onde podemos utilizar conhecimento adquirido dentro de sala de aula. A figura a seguir demostra a página de configuração da PythonAnywhere, que é onde você precisará de ir quando quiser fazer alterações no servidor. A programação é baseada nas linguagens *python*, *SQL* e *html,* no Apêndice B, estão as principais programações feitas para funcionamento do servidor.

Figura 4.4: Pagina de configuração da pythonanywhere



Fonte: autoria própria, 2017

O Sistema de supervisão e controle por *tablets* e *smartphones* desenvolvido no *mit app inventor* tem um custo de aquisição gratuita, que fornece ao usuário monitoramento e controle remoto de qualquer dispositivo microcontrolado.

Ao desenvolver o aplicativo levou-se em consideração que este deveria exibir de forma clara e direta os dados através de uma interface amigável e simples.

4.4 Esquema elétrico do multiplexador

Para se ligar dois sensores no microcontrolador wemos, é necessário utilizar um circuito integrado para multiplexar a porta analógica, uma vez que o mesmo possui apenas uma. A figura 4.5 mostra o circuito multiplexador montado para leitura

de tensão e corrente, onde o wem1, adm1, mod2, mod1 são respectivamente o microcontrolador wemos, multiplexador, sensor de corrente e tensão.

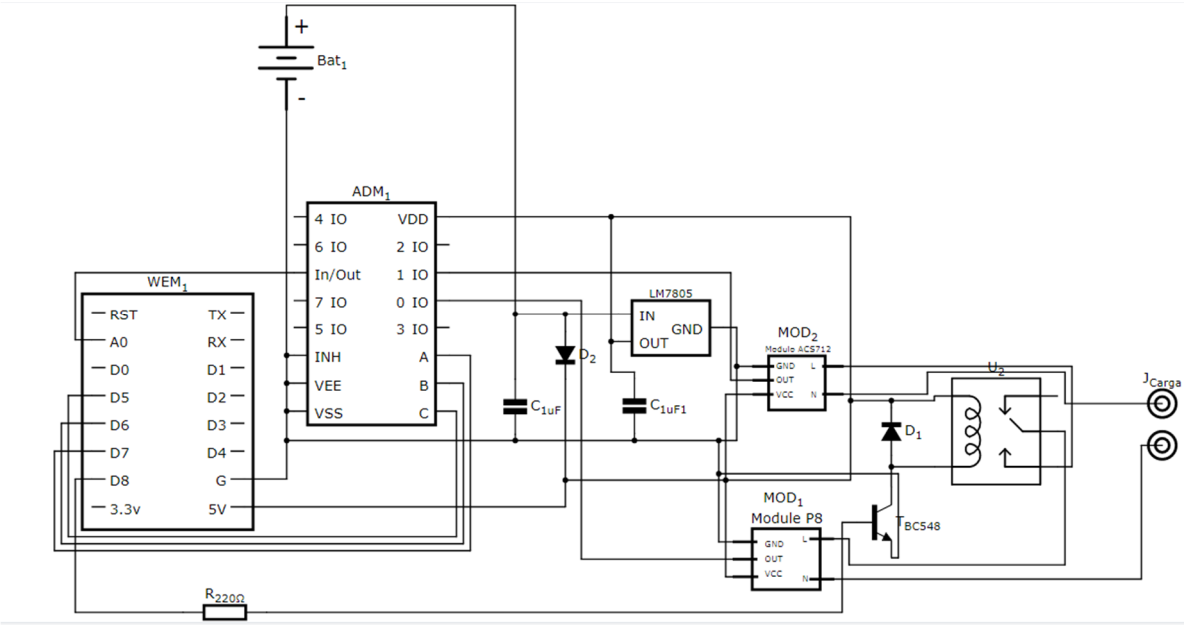


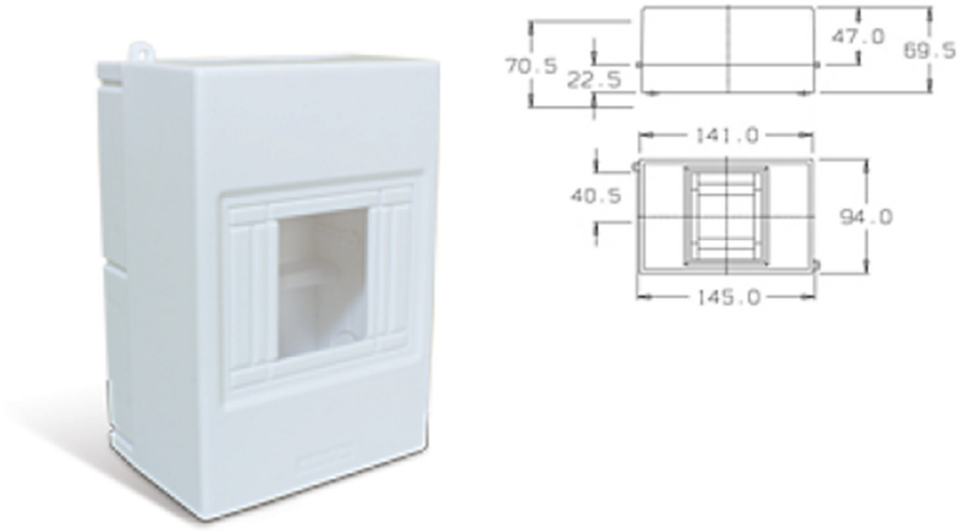
Figura 4.5: Circuito elétrico

Ferramenta utilizada: 123d circuits.

4.5 Desenho mecânico da caixa

A figura 4.6 mostra as dimensões da caixa usada para a montagem do protótipo.

Figura 4.6: Desenho Mecânico da Caixa do Dispositivo



Fonte:http://www.steck.com.br/assets/uploads/produto_categorias/197/documento/b247573886f06a64bb684e0d97fbf90d.pdf

4.6 Montagem do protótipo

O protótipo do dispositivo foi construído para controlar uma carga especifica e analisar o valor de corrente e tensão considerando um circuito monofásico ou bifásico, através destes dados foi possível realizar uma programação para chegar no

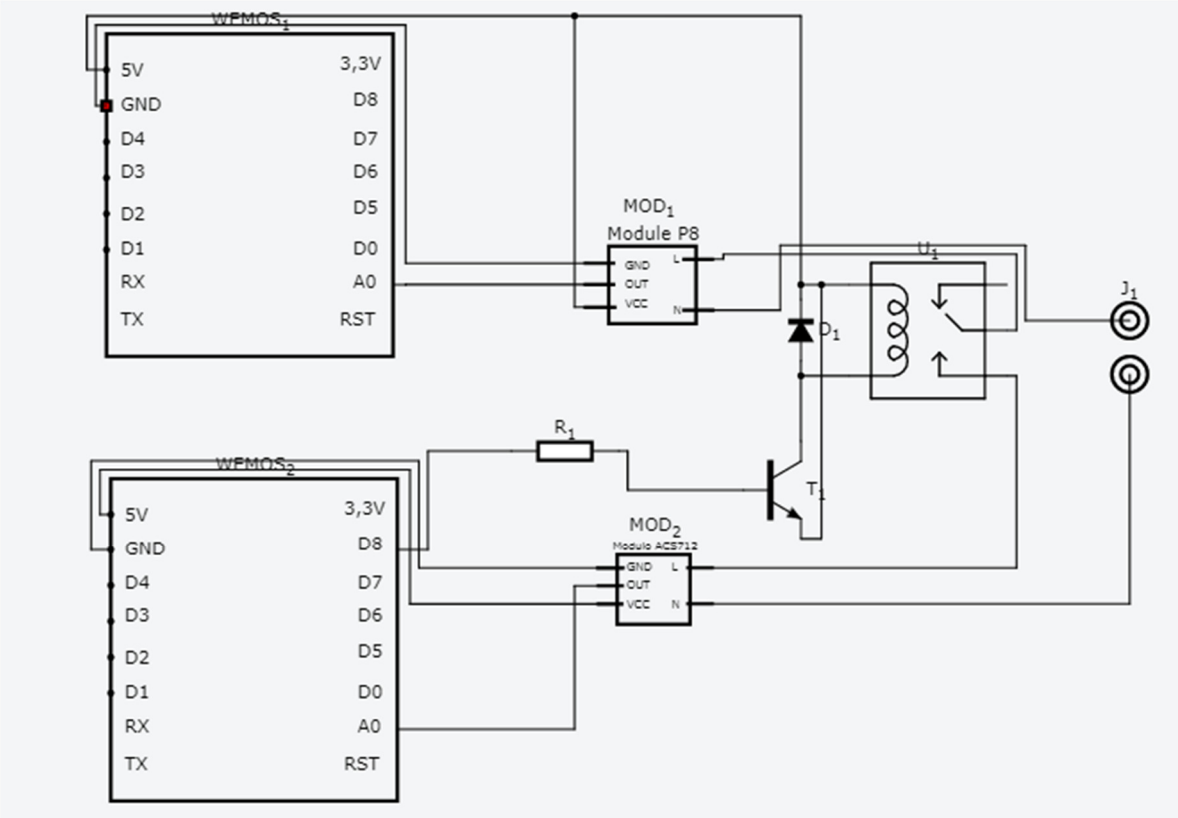
cálculo da potência do equipamento ligado em um ponto de tomada especifico. Os cálculos realizados pelo wemos são simples, para conhecer a corrente que o circuito está consumindo, a tensão da rede e da potência do equipamento. Na figura 4.7 é apresentado o protótipo finalizado.

Figura 4.7: protótipo finalizado

Fonte: autoria própria, 2017

Na figura a seguir podemos observar o circuito do protótipo finalizado.

Figura 4.8: circuito do dispositivo



Fonte: autoria própria, 2017

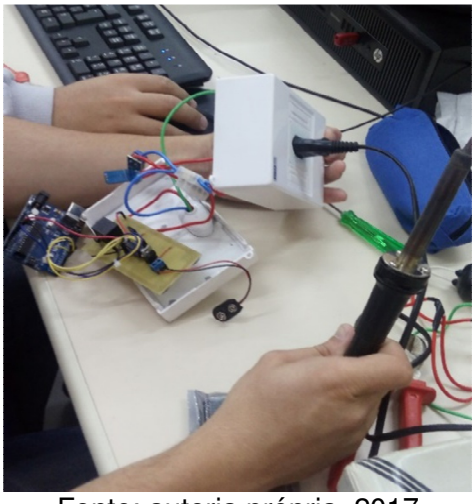
5 AVALIAÇÃO DOS RESULTADOS E CONSIDERAÇÕES FINAIS

Neste capitulo é apresentado os resultados dos testes realizados no protótipo, controlando uma carga de forma remota, coletando amostras e comparando com um multímetro.

5.1 Avaliações dos resultados

Para que fosse possível verificar se os resultados obtidos no protótipo estavam corretos, foi realizado medições com outros equipamentos, com o objetivo de realizar as comparações dos valores encontrados no protótipo com valores obtidos pelo dispositivo de medição de grandezas elétricas. Foram realizados ensaios para verificar a veracidade das medidas obtidas pelo protótipo, neste ensaio foi utilizado uma carga resistiva conhecida como lâmpada incandescente. A figura 5.1 mostra a bancada de teste para realizar os testes de leitura na lâmpada incandescente.

Figura 5.1: bancada de teste



Fonte: autoria própria, 2017

No início, para realizar a medição de corrente, obteve-se valores muito acima do especificado para o equipamento e, após realizado testes, foi verificado que para este módulo com o modelo de +30A -30A sua calibração para a sensibilidade seria de 66mA, esse valor foi utilizado para calcular a corrente do circuito, desenvolvido conforme Apêndice A.

Figura 5.2: Código para medição e calibração do sensor de corrente.

```
void loop() {
  digitalWrite(D7, HIGH);
  corrente = 0;
  for (int i = 10000; i > 0; i--) {
    sensorValue_aux = (analogRead(pinoSensor) - 756);
    valorSensor += pow(sensorValue_aux, 2);
    delay(1);
  }
  Serial.print("sub: ");
  Serial.println(sensorValue_aux);
  valorSensor = (sqrt(valorSensor / 10000)) * voltsporUnidade;
  corrente = (valorSensor / sensibilidade);
  if (corrente <= 0.15) {
    corrente = 0;
  }
  Serial.println(corrente);
  Serial.print("sensor: ");
  Serial.println(analogRead(pinoSensor));
  valorSensor = 0;
```

Fonte: autoria própria, 2017. Ferramenta: Arduíno (disponível em Apêndice A)

Para que os dados fossem enviados ao servidor, primeiro configuramos a rede *wifi* no qual o *wemos* foi interligado:

Figura 5.3: Código exemplo para conexão *wifi.*

Fonte: autoria própria, 2017.Ferramenta: Arduíno (disponível em Apêndice A)

```
void setup()
{
  Serial.begin(9600);
  delay(100);
  pinMode(pinoSensor, INPUT);
  pinMode(D7, OUTPUT);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println("ESP8266 in sleep mode");
}
```

Após realizado a conexão o meio utilizado para o envio dos dados foi através da estrutura JSON, um formato de padrão aberto que utiliza texto legível a humanos para transmitir objetos de dados consistindo de pares atributo-valor.

Figura 5.4: Código exemplo para publicação dos dados no servidor *pythonanywhere.*

Fonte: autoria própria, 2017. Ferramenta: Arduíno (disponível em Apêndice A)

Assim como esta estrutura também foi responsável pela recepção de dados do servidor para determinar se a carga deve estar ligada ou desligada:

Figura 5.5: Código exemplo para obter o estado da carga do servidor.

```
void GET(void){
  delay(500);
  int estado;
  String line;
   int state_pos;
   Serial.print("connecting to ");
  Serial.println(host);

  WiFiClient client;
  const int httpPort = 80;

  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }


  Serial.print("Requesting GET: ");
  client.println("GET /voltadado HTTP/1.1");
  client.println("Host: smartja.pythonanywhere.com");
  client.println("Connection: close");
  client.println();
  delay(500);
  while (client.available()) {
  line = client.readStringUntil('\r');
  Serial.println(line);
```

```
  state_pos = line.length();
  Serial.println(state_pos);
  }
    if (state_pos == 10)
       estado = 0;
    if (state_pos == 7)
       estado = 1;

    if (estado == 1){
      Serial.println("LED ON");
      digitalWrite(relePin, HIGH);}
    if (estado == 0){
      Serial.println("LED OFF");
      digitalWrite(relePin, LOW);}
      Serial.println(estado);


    }
```

Fonte: autoria própria, 2017. Ferramenta: Arduíno (disponível em Apêndice A)

No servidor foi utilizada a linguagem SQL para armazenar os valores em um banco de dados disponibilizado pelo próprio servidor, e para a amostragem destes dados no aplicativo de monitoramento:

Figura 5.6: Código exemplo para armazenar os valores no banco de dados, e para utilizá-los na tela de supervisão.

```
37
38  @app.route('/upload', methods=['POST','GET'])
39  def upar():
40
41      data= request.get_json()
42
43      tensao=data['tensao']
44      corrente=data['corrente']
45
46      if data['corrente'] == '0':
47          estado = 'Desligado'
48
49
50      conn = sqlite3.connect('o.db')
51      cursor = conn.cursor()
52      cursor.execute("""
53      UPDATE clientes
54      SET tensao = ?, corrente = ?
55              """, (tensao, corrente))
56
57      conn.commit()
58
59
60      cursor.execute("""
61      UPDATE clientes1
62      SET estado = ?, carga = ?
63              """, (estado, "Carga1"))
64      conn.commit()
65      conn.close()
66
```

```
68  @app.route('/teste', methods=['POST','GET'])
69  def criar_monitoramento():
70
71      conn = sqlite3.connect('o.db')
72      cursor = conn.cursor()
73
74      cursor.execute("""
75      SELECT * FROM adriano;
76      """)
77
78      texto_retornos = []
79      for linhas in cursor.fetchall():
80          listas = []
81          for items in linhas:
82              listas.append(items)
83      texto_retornos.append(linhas)
84
85      cursor.execute("""
86      SELECT * FROM clientes;
87      """)
88
89      texto_retorno = []
90      for linha in cursor.fetchall():
91          lista = []
92          for item in linha:
93              lista.append(item)
94      texto_retorno.append(linha)
95      conn.close()
```

Fonte: autoria própria, 2017. Ferramenta: *Website pythonanywhere* (Apêndice B)

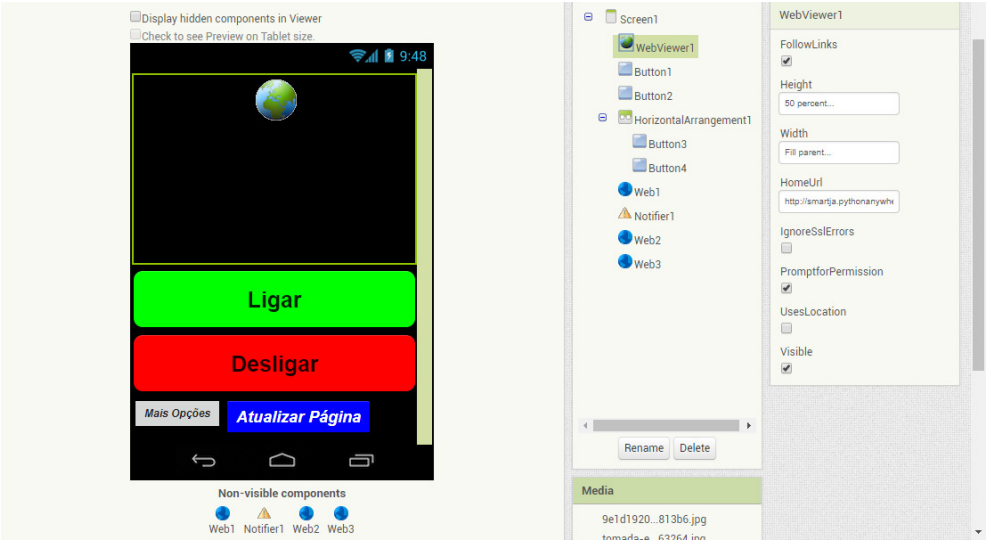
Para que os dados fossem entregues ao usuário, utilizamos a linguagem HTML para a criação de um site, e em seguida configuramos no desenvolvedor do aplicativo para que o mesmo mostrasse este site com os dados.

Figura 5.7: Código exemplo para criação do site de supervisão.

```
pagina = """<!DOCTYPE html>
    <html>
     <title>Valores Obtidos</title>

        <head>

        <meta http-equiv="refresh" content="3" >
        </head>

            <body style="background-color:black">

                <div style="border: 2px solid;border-radius: 25px;box-shadow: 10px 10px 5px #00ffff; background-color:#009999">
                    <h2 style="border: 2px solid;border-radius: 15px;background-color:white;font-size:25px;color:black;text-align:center">Tomada:</h2>
                        <h1 style="border: 2px solid;border-radius: 15px;background-color:#003333;font-size:25px;color:white;text-align:center">{0}</h1></div>

                <div style="border: 2px solid;border-radius: 25px;box-shadow: 10px 10px 5px #66ff99; background-color:#009933">
                    <h2 style="border: 2px solid;border-radius: 15px;background-color:white;font-size:25px;color:black;text-align:center">Estado:</h2>
                        <h1 style="border: 2px solid;border-radius: 15px;background-color:#003300;font-size:25px;color:white;text-align:center">{1}</h1></div>

                <div style="border: 2px solid;border-radius: 25px;box-shadow: 10px 10px 5px #4db8ff; background-color:#0047b3">
                    <h2 style="border: 2px solid;border-radius: 15px;background-color:white;font-size:25px;color:black;text-align:center">Tensão:</h2>
                        <h1 style="border: 2px solid;border-radius: 15px;background-color:#000066;font-size:25px;color:white;text-align:center">{2}V</h1></div>

                <div style="border: 2px solid;border-radius: 25px; box-shadow: 10px 10px 5px #ff471a;background-color:#660000">
                    <h2 style="border: 2px solid;border-radius: 15px;background-color:white;font-size:25px;color:black;text-align:center">Corrente:</h2>
                        <h1 style="border: 2px solid;border-radius: 15px;background-color:#1a0000;font-size:25px;color:white;text-align:center">{3}A</h1></div>

        </body>
```

Fonte: autoria própria, 2017. Ferramenta: *Website pythonanywhere* (Apêndice B).

Figura 5.8: Imagem da criação do aplicativo.



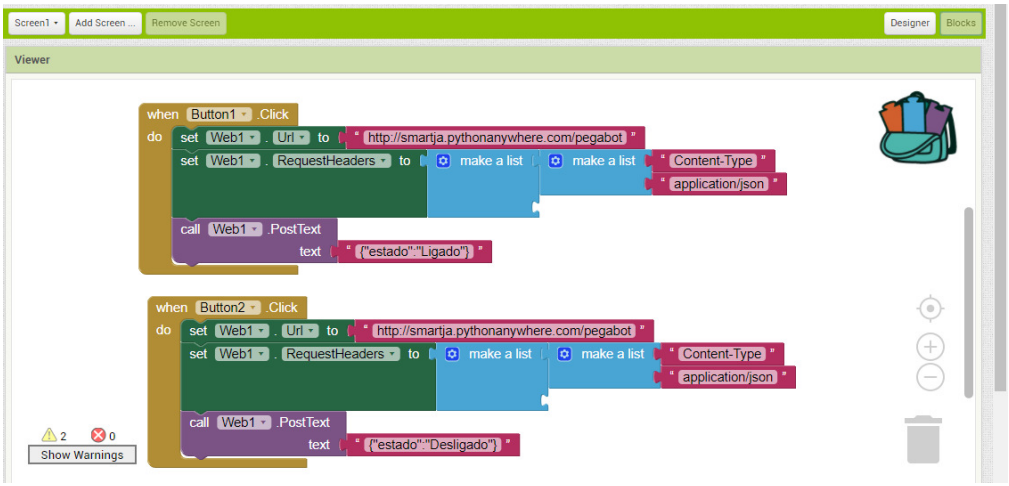
Fonte: autoria própria, 2017. Ferramenta: *Website MIT App Inventor*.

Foi criado também no aplicativo os botões para a energização ou desenergização da carga, setar os valores máximos em que a carga possa atingir, e determinar em quanto tempo pudesse ser feito sua energização ou desenergização.

No próprio site desenvolvedor do aplicativo, foi utilizado uma programação em blocos que facilitou para que pudesse estar enviando os dados ao servidor (também

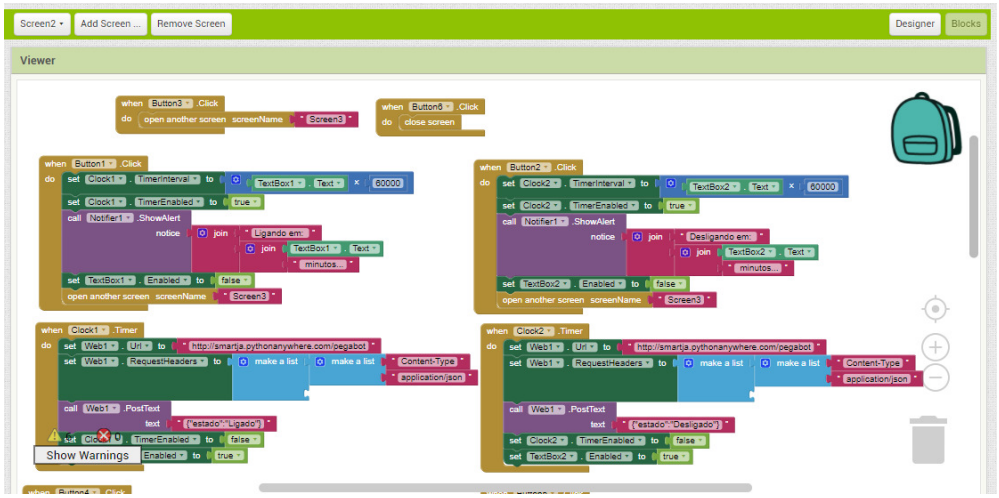
em formato JSON), para que em seguida fosse enviado à plataforma *wemos* e realizando o controle da carga.

Figura 5.9: Imagem da criação do aplicativo, lógica em blocos da primeira tela.



Fonte: autoria própria, 2017. Ferramenta: *Website MIT App Inventor*.

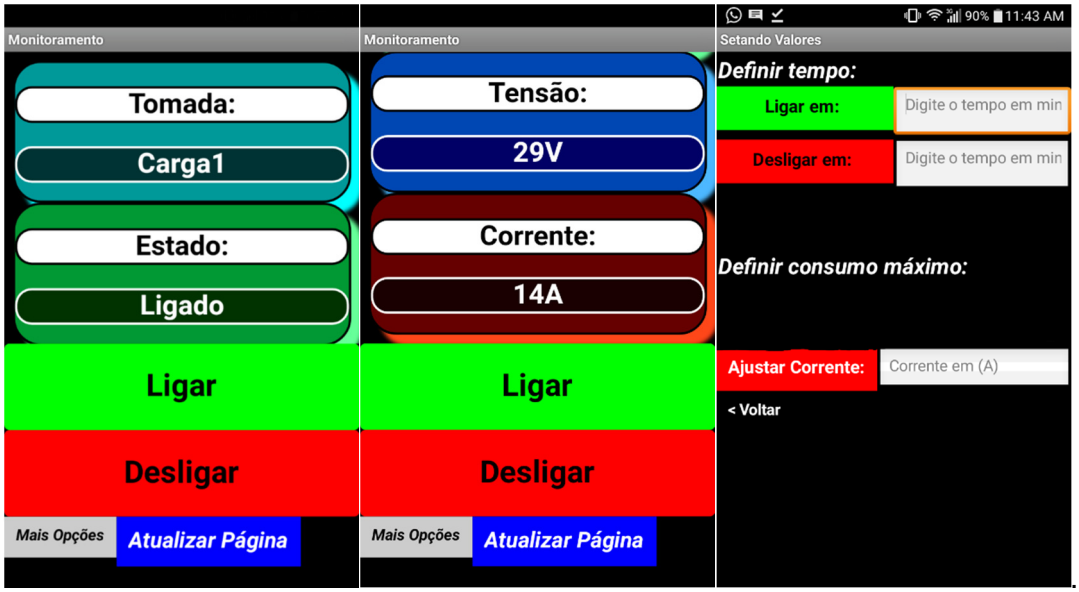
Figura 5.10: Imagem da criação do aplicativo, lógica em blocos da segunda tela.



Fonte: autoria própria, 2017. Ferramenta: *Website MIT App Inventor*.

Ao finalizado todas as telas e a lógica para “interligar” o usuário ao servidor, obteve-se o seguinte aplicativo:

Figura 5.11: Aplicativo de supervisão e controle, já instalado no *smartphone*.



Fonte: autoria própria, 2017

Após os ajustes realizados em nosso dispositivo, é possível observar que os resultados apresentados são satisfatórios, pois o valor obtido pelo sensor apresenta um erro que está em média 28%. No apêndice C, apresenta valores obtidos na realização de alguns testes para verificar a eficácia do protótipo.

5.2 Considerações finais

Medição de tensão: Após realizados testes, foi descoberto que o sensor de tensão AC P8 é composto por um fotoacoplador, na onde o mesmo é responsável apenas pela detecção da tensão que passa sobre ele, mas não determina o valor exato. Para o projeto desenvolvido, o valor de tensão em que foi desejado medir foi o fornecido pela rede (127V), sendo assim quando o equipamento estiver desenergizado, o microcontrolador receberá o valor 0, e quando energizado receberá o valor de 127V. Uma solução para medição tensão é utilizar um transformador 127V AC/ 0-5V DC.

Multiplexação de IO's: O grupo obteve insucesso na multiplexação da única porta analógica em que a plataforma *wemos* possui para que houvesse a medição dos dois sensores na única plataforma. Após realização de alguns testes, observamos que o multiplexador estava interferindo na leitura dos valores dos sensores, por esta questão foi optado pela utilização de um outro microcontrolador *wemos,* sendo um responsável pela medição de corrente e o envio deste valor para o servidor, e o outro responsável medição de tensão ,o envio do valor para o servidor e o controle do relé para o controle da carga. Uma outra solução para este caso, seria a utilização de uma outra plataforma com mais IO's analógicas.

6 CONCLUSÕES E DESENVOLVIMENTOS FUTUROS

Este trabalho atingiu os objetivos específicos, de desenvolver um dispositivo que possa ser controlado e supervisionado a distância, onde ler o consumo de corrente em tempo real, detecta a tensão e com base nos dois valores anteriores estima a potência, com custo acessível. O sistema supervisório do dispositivo facilita a visualização e o controle remoto dos dados de um determinado aparelho, possibilitando uma melhor praticidade e segurança no acompanhamento da carga, podendo ser usado para reduzir o desperdício de energia. Junto ao sistema supervisório o usuário poderá controlar o estado e monitorar parâmetros de tensão, corrente e potência de um determinado equipamento.

A solução adotada, com o sensor hall para detecção de corrente, o módulo sensor de tensão AC P8 e o microcontrolador com modulo ESP 8266 integrado, se apresentou com o melhor custo benefício. Este protótipo apresentou uma performance confiável, com baixo custo de implementação e facilidade de manuseio, assim como a facilidade da visualização dos dados em um aplicativo. Os objetivos foram atingidos com êxito. O microcontrolador *wemos* funcionou conforme o esperado e através dele, dos sensores e do relé foi possível controlar estado da carga e medir com eficiência valores de energia elétrica.

O projeto apresenta uma visão voltada para a conectividade, possibilitando a um usuário comum controlar e monitorar um determinado aparelho a distância. Futuramente, é pretendido realizar melhorias no protótipo a fim de que o dispositivo possa fazer o cálculo de demanda onde poderá ser acompanhado em tempo real, com isso, será disponibilizado ao usuário um controle mais sofisticado do consumo de energia.

REFERÊNCIAS BIBLIOGRÁFICAS

ADEMARO A. M. B. COTRIM, Instalações .Elétricas. 5a. Ed .2009

ANDREW CLARK, App Inventor launches second iteration, 2013. Disponível em: < http://news.mit.edu/2013/app-inventor-launches-second-iteration>

BRENO S. LEMOS, GLEYDSON I. BARBOSA, JORGE F. M. C. SILVA, J. W. M. MENEZES E THIAGO Q. DE OLIVEIRA, Estudo Comparativo de Middlewares para Internet das Coisas usando Plataformas Livres no Monitoramento de Ambientes. Disponivel em: <http://sbrt.org.br/sbrt2016/anais/IC02/1570281029.pdf>

CARLOS ALBERTO KAMIENSKI e DJAMEL SADOK, Qualidade de serviço na internet. Disponível em: <https://www.researchgate.net/profile/Carlos_Kamienski/publication/242087340_Qualidade_de_Servico_na_Internet/links/0046352f7f0ffe4664000000.pdf>

DAVE EVANS, The Internet of Things How the Next Evolution of the Internet Is Changing Everything, 2011.

Espressif, Version 5.4, disponível em: <http://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf> [2], acesso em: 10/05/2017

FERNANDO NUNES BELCHIOR, Medidas Elétricas, 2016. Disponível em: <http://www.gqee.unifei.edu.br/arquivos_upload/disciplinas/31/Apostila%20ELE505%20-%20Medidas%20Eletricas.pdf> Acesso em: 20/05/2017.

FERNANDO SILVA OZUR, THIAGO HENRIQUE PEREIRA, JOANA D'ARQUE DA SILVA CORREA, Controle de demanda de energia elétrica. Disponível em:

<http://revistas2.unibh.br/index.php/dcet/article/view/696/405>

Acesso em :24/05/2017

http://ensinodematemtica.blogspot.com.br/2010/11/campo-magnetico-condutor-retilineo.html

https://forum.arduino.cc/index.php?topic=283043.0 [3]

http://ijetst.in/article/v3-i11/3%20ijetst.pdf [4]

Jim Chase, The Evolution of the Internet of Things, 2013.

KOLBAN, N. Kolban's Book on ESP8266. 2016.

LÍVIA CUNHA, Relés e Contadores. Disponível em: <http://www.instalacoeseletricas.com/download/Radiografia_reles_contatores_out09.pdf>

LUCIANA GUIMARÃES CARVALHO**,** Características da utilização de software de código aberto: Um estudo sobre o setor de tecnologia da informação, 2013 Disponível em:

<http://www.fumec.br/revistas/sigc/article/viewFile/1931/1227>
Acesso em: 22/05/2017

MARCELLO A. GÓMEZ MAUREIRA, DAAN OLDENHOF, LIVIA TEERNSTRA, ThingSpeak – an API and Web Service for the Internet of Things. Disponivel em: https://staas.home.xs4all.nl/t/swtr/documents/wt2014_thingspeak.pdf, Acesso em: 22/05/2017

PAULO R. P. CEZAR JR., JEAN-JACQUES DE GROOTE, Automação residencial com microcontroladores Arduino e controle via internet para cursos de engenharia, 2015. Disponível em: <http://copec.eu/congresses/wcseit2015/proc/works/17.pdf>

ROBERT. L. BOYLESTAD, Introdução à analise de circuitos 10ª Ed., 2004.

TANCICLEIDE C. S. GOMES, JEANE C. B. DE MELO, App Inventor for Android: Uma Nova Possibilidade para o Ensino de Lógica de Programação. Disponível em: <http://www.br-ie.org/pub/index.php/wcbie/article/view/2725/2379>

TEXAS INSTRUMENTS, CD4051B, CD4052B, CD4053B, disponível em:

<http://www.ti.com/lit/ds/symlink/cd4051b.pdf>

VAGNER GUADAGNIN E JÚLIO CÉSAR MARQUES DE LIMA, Projeto e implementação de um protótipo de registrador de grandezas elétricas.

YHDC , Split core current transformer, disponível em :

https://s3-sa-east-1.amazonaws.com/robocore-lojavirtual/831/Datasheet_SCT013.pdf [5], acesso em: 11/05/2017

REFERÊNCIAS DE FIGURAS

Figura 2.1 – http://www.eletrica.ufpr.br/edu/Sensores/2000/neis/, Acesso em: 05/2017.

Figura 2.2 – http://www.wikipremed.com/image.php?img=010403_68zzzz141800_Electromagnetism_68.jpg&image_id=141800, Acesso em: 05/2017.

Figura 2.3 – http://www.ebah.com.br/content/ABAAAAQDoAC/eletricidade, Acesso em: 2017.

Figura 2.4 – DAVE EVANS, The Internet of Things How the Next Evolution of the Internet Is Changing Everything, 2011.

Figura 2.6 – http://androidcontrol.blogspot.com.br/2015/06/android-iot-control-thingspeak.html, Acesso em 05/2017.

Figura 2.7 – https://blog.butecopensource.org/tag/esp8266/, Acesso em 05/2017.

Figura 2.8 – http://www.instructables.com/id/Programming-the-WeMos-Using-Arduino-SoftwareIDE/, Acesso em 05/2017.

Figura 2.10 – https://playground.arduino.cc/Learning/4051, 2017.

Figura 2.11 – appinventor.googlelabs.com/ode/ya.html#1222163, 2017.

Figura 2.12 – http://appinventor.mit.edu/explore/designer-blocks.html, 2017.

Figura 4.1 – www.confrariadaescala.com.br/modulo-sensor-de-tensao-ac-p8-002014-gbk-robotics.html, 2017.

Figura 4.2 – https://eletronicos.mercadolivre.com.br/pecas-e-componentes-eletricos-sensores/sensor-de-corrente-acs714, 2017.

Figura 4.3 – http://www.buscape.com.br/roteador, 2017.

Figura 4.6 – http://www.steck.com.br/assets/uploads/produto_categorias/197/documento/b247573886f06a64bb684e0d97fbf90d.pdf.

Apêndice A

CÓDIGO PARA O *WEMOS* MEDIDOR DE CORRENTE:

```
#include <math.h>
#include <ESP8266WiFi.h>
const char* ssid     = "smartja";
const char* password = "smartjar";
String pubString;
const char* host = "smartja.pythonanywhere.com";
const int sleepTimeS = 1 * 10;

float pinoSensor = A0;
float corrente = 0;
float sensorValue_aux = 0;
float valorSensor = 0;
float valorCorrente = 0;
float voltsporUnidade = 0.004887586;// 5%1023
float sensibilidade = 0.066;

void setup()
{
  Serial.begin(9600);
  delay(100);
  pinMode(pinoSensor, INPUT);
  pinMode(D7, OUTPUT);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println("ESP8266 in sleep mode");
}

void loop() {
  digitalWrite(D7, HIGH);
  corrente = 0;
  for (int i = 10000; i > 0; i--) {
    sensorValue_aux = (analogRead(pinoSensor) - 756);
    valorSensor += pow(sensorValue_aux, 2);
    delay(1);
  }
  Serial.print("sub: ");
  Serial.println(sensorValue_aux);
  valorSensor = (sqrt(valorSensor / 10000)) * voltsporUnidade;
  corrente = (valorSensor / sensibilidade);
  if (corrente <= 0.095) {
    corrente = 0;
  }
  Serial.println(corrente);
  Serial.print("sensor: ");
  Serial.println(analogRead(pinoSensor));
  valorSensor = 0;

```

```
  POST();

}
void POST(void)
{
  delay(500);
  Serial.print("connecting to ");
  Serial.println(host);

  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }
  pubString = "{\"corrente\": " + String(corrente) + "}";

  String pubStringLength = String(pubString.length(), DEC);

  Serial.print("Requesting POST: ");
  client.println("POST /uploads HTTP/1.1");
  client.println("Host: smartja.pythonanywhere.com");
  client.println("Content-Type: application/json");
  client.println("Connection: close");
  client.print("Content-Length: ");
  client.println(pubStringLength);
  client.println();
  client.print(pubString);
  client.println();
  delay(500);

  while (client.available()) {
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }
  Serial.println();
  Serial.println("closing connection");
}
```

CÓDIGO PARA O *WEMOS* MEDIDOR DE TENSÃO:

```
100
101  #include <math.h>
102  #include <ESP8266WiFi.h>
103
104  const char* ssid     = "smartja";
105  const char* password = "smartjar";
106  String pubString;
107  const char* host = "smartja.pythonanywhere.com";
108  const int sleepTimeS = 1*10;
109
110  float tensao = 0;
111  const int relePin = D7;
112
113  void setup()
114  {
115    pinMode(relePin,OUTPUT);
116    Serial.begin(9600);
117    delay(100);
118    Serial.println();
119    Serial.print("Connecting to ");
120    Serial.println(ssid);
121    WiFi.begin(ssid, password);
122
123    while (WiFi.status() != WL_CONNECTED) {
124      delay(500);
125      Serial.print(".");
126    }
127    Serial.println("");
128    Serial.println("WiFi connected");
129    Serial.println("IP address: ");
130    Serial.println(WiFi.localIP());
131    Serial.println("ESP8266 in sleep mode");
132  }
133  void loop(){
134
135      tensao = (analogRead(0)*127)/1023;
136      if(tensao<10)
137      tensao=0;
138      Serial.print("Tensao: ");
139      Serial.print(tensao);
140      Serial.println("V");
141       delay(1000);
142    POST();
143    GET();
144   }
145     void POST(void)
146   {
147     delay(500);
148     Serial.print("connecting to ");
149     Serial.println(host);
150
```

```
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }
  pubString = "{\"tensao\": "+ String(tensao) +"}";

  String pubStringLength = String(pubString.length(), DEC);

  Serial.print("Requesting POST: ");
  client.println("POST /upload HTTP/1.1");
  client.println("Host: smartja.pythonanywhere.com");
  client.println("Content-Type: application/json");
  client.println("Connection: close");
  client.print("Content-Length: ");
  client.println(pubStringLength);
  client.println();
  client.print(pubString);
  client.println();
  delay(500);
  while (client.available()) {
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }
  Serial.println();
  Serial.println("closing connection");
}

void GET(void){
  delay(500);
  int estado;
  String line;
   int state_pos;
   Serial.print("connecting to ");
  Serial.println(host);

  WiFiClient client;
  const int httpPort = 80;

   if (!client.connect(host, httpPort)) {
     Serial.println("connection failed");
     return;
   }
   Serial.print("Requesting GET: ");
   client.println("GET /voltadado HTTP/1.1");
   client.println("Host: smartja.pythonanywhere.com");
   client.println("Connection: close");
   client.println();
   delay(500);
   while (client.available()) {
   line = client.readStringUntil('\r');
   Serial.println(line);
   state_pos = line.length();
   Serial.println(state_pos);
   }
     if (state_pos == 10)
        estado = 0;
     if (state_pos == 7)
        estado = 1;

     if (estado == 1){
       Serial.println("LED ON");
       digitalWrite(relePin, HIGH);}
     if (estado == 0){
       Serial.println("LED OFF");
       digitalWrite(relePin, LOW);}
       Serial.println(estado);
     }

```

Apêndice B

CÓDIGO PARA O SERVIDOR *WEB PYTHONANYWHERE*:

```python
from flask import Flask, request
import sqlite3

app = Flask(__name__)

@app.route('/reload', methods=['POST','GET'])
def criarbd():
    conn = sqlite3.connect('o.db')
    cursor = conn.cursor()

    cursor.execute ("""
        CREATE TABLE table1 (

        estado VARCHAR(9) NOT NULL,
        carga VARCHAR(9) NOT NULL
        );""")

    cursor.execute("""
        INSERT INTO table1 (estado, carga)
        VALUES (?,?)
        """, ('5','Carga1'))

    cursor.execute ("""
        CREATE TABLE clientes (

        tensao VARCHAR,
        corrente VARCHAR
        );""")

    cursor.execute("""
    INSERT INTO clientes (tensao, corrente)
    VALUES (?,?)
    """, ('124','1'))

    conn.commit()
    conn.close()

@app.route('/upload', methods=['POST','GET'])
def upar():

    data= request.get_json()

    tensao=data['tensao']
    corrente=data['corrente']

    if data['corrente'] == '0':
        estado = 'Desligado'


    conn = sqlite3.connect('o.db')
    cursor = conn.cursor()
    cursor.execute("""
    UPDATE clientes
    SET tensao = ?, corrente = ?
            """, (tensao, corrente))

    conn.commit()


    cursor.execute("""
    UPDATE adriano
    SET estado = ?, carga = ?
            """, (estado, "Carga1"))
    conn.commit()
    conn.close()


@app.route('/teste', methods=['POST','GET'])
def criar_monitoramento():

    conn = sqlite3.connect('o.db')
    cursor = conn.cursor()

    cursor.execute("""
    SELECT * FROM adriano;
    """)
```

```
    texto_retornos = []
    for linhas in cursor.fetchall():
        listas = []
        for items in linhas:
            listas.append(items)
    texto_retornos.append(linhas)

    cursor.execute("""
    SELECT * FROM clientes;
    """)

    texto_retorno = []
    for linha in cursor.fetchall():
        lista = []
        for item in linha:
            lista.append(item)
    texto_retorno.append(linha)
    conn.close()

    pagina = """<!DOCTYPE html>
        <html>
         <title>Valores Obtidos</title>

            <head>

            <meta http-equiv="refresh" content="3" >
            </head>

                <body style="background-color:black">
<div style="border: 2px solid;border-radius: 25px;box-shadow: 10px 10px 5px #00ffff; background-color:#009999">
<h2 style="border: 2px solid;border-radius: 15px;background-color:white;font-size:25px;color:black;text-align:center">Tomada:</h2>
<h1 style="border: 2px solid;border-radius: 15px;background-color:#003333;font-size:25px;color:white;text-align:center">{0}</h1></div>

<div style="border: 2px solid;border-radius: 25px;box-shadow: 10px 10px 5px #66ff99; background-color:#009933">
<h2 style="border: 2px solid;border-radius: 15px;background-color:white;font-size:25px;color:black;text-align:center">Estado:</h2>
<h1 style="border: 2px solid;border-radius: 15px;background-color:#003300;font-size:25px;color:white;text-align:center">{1}</h1></div>

<div style="border: 2px solid;border-radius: 25px;box-shadow: 10px 10px 5px #4db8ff; background-color:#0047b3">
<h2 style="border: 2px solid;border-radius: 15px;background-color:white;font-size:25px;color:black;text-align:center">Tensão:</h2>
<h1 style="border: 2px solid;border-radius: 15px;background-color:#000066;font-size:25px;color:white;text-align:center">{2}V</h1></div>

<div style="border: 2px solid;border-radius: 25px; box-shadow: 10px 10px 5px #ff471a;background-color:#660000">
<h2 style="border: 2px solid;border-radius: 15px;background-color:white;font-size:25px;color:black;text-align:center">Corrente:</h2>
<h1 style="border: 2px solid;border-radius: 15px;background-color:#1a0000;font-size:25px;color:white;text-align:center">{3}A</h1></div>

            </body>

        </html>
    """.format(texto_retornos[0][1],texto_retornos[0][0],texto_retorno[0][0],texto_retorno[0][1])
    return  pagina

@app.route('/pegabot', methods=['POST','GET'])
def pegabot():
```

```
    data= request.get_json()
    estado  = data['estado']

    conn = sqlite3.connect('o.db')
    cursor = conn.cursor()
    cursor.execute("""
    UPDATE table1
    SET estado = ?,carga = ?
            """, (estado,"Carga1"))
    conn.commit()
    conn.close()


@app.route('/voltadado', methods=['POST','GET'])
def voltaprowemos():

    conn = sqlite3.connect('o.db')
    cursor = conn.cursor()

    cursor.execute("""
    SELECT * FROM table1;
    """)

    texto_retornos = []
    for linhas in cursor.fetchall():
        listas = []
        for items in linhas:
            listas.append(items)
    texto_retornos.append(linhas)


    return texto_retornos[0]

    conn.close()

@app.route('/setat', methods=['POST','GET'])
def setatensao():

        data = request.get_json()
        tensao = data['tensao']
        conn = sqlite3.connect('o.db')
        cursor = conn.cursor()
        cursor.execute("""
        SELECT * FROM clientes;
        """)

        texto_retorno = []
        for linha in cursor.fetchall():
         lista = []
         for item in linha:
            lista.append(item)
            texto_retorno.append(linha)

            if float(texto_retorno[0][0]) > float(tensao):
                estado = 'Desligado'

                cursor.execute("""
                UPDATE table1
                SET estado = ?,carga = ?
                """, (estado,"Carga1"))

        conn.commit()
        conn.close()

@app.route('/setac', methods=['POST','GET'])
def setacorrente():

        data = request.get_json()
        corrente = data['corrente']
        conn = sqlite3.connect('o.db')
        cursor = conn.cursor()
        cursor.execute("""
        SELECT * FROM clientes;
        """)
```

```
207
208             texto_retorno = []
209             for linha in cursor.fetchall():
210              lista = []
211              for item in linha:
212                  lista.append(item)
213                  texto_retorno.append(linha)
214
215                  if float(texto_retorno[0][1]) > float(corrente):
216                      estado = 'Desligado'
217
218              cursor.execute("""
219              UPDATE table1
220              SET estado = ?,carga = ?
221              """, (estado,"Carga1"))
222              conn.commit()
223              conn.close()
224
225  if __name__ == '__main__':
226      app.run()
```

Apêndice C

Teste medição de corrente de uma lâmpada incandescente		
Corrente (A)		
Valor teórico	Protótipo	Erro
0,47	0,48	34%
0,47	0,481	34,1%
0,47	0,475	33,6%
0,47	0,476	33,7%

Teste medição de corrente do ferro de solda		
Corrente (A)		
Valor teórico	Protótipo	Erro
0,47	0,473	33,5
0,47	0,47	0%
0,47	0,471	33,4%
0,47	0,475	33,6%

Teste medição de corrente do ventilador		
Corrente (A)		
Valor teórico	Protótipo	Erro
0,48	0,482	33,4%
0,48	0,48	0%
0,48	0,483	33,5%
0,48	0,484	33,6%