INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA – CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE TECNOLOGIA EM ELETRÔNICA INDUSTRIAL

HERMESON JEREMIAS

ESTUDO DE UM MEDIDOR DE CORRENTE CONECTADO COM UMA PLATAFORMA IOT

FLORIANÓPOLIS, 2020.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA – CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE TECNOLOGIA EM ELETRÔNICA INDUSTRIAL

HERMESON JEREMIAS

ESTUDO DE UM MEDIDOR DE CORRENTE CONECTADO COM UMA PLATAFORMA IOT

Trabalho de Conclusão de Curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de Tecnólogo em Eletrônica Industrial.

Orientador:
Prof. Samir Bonho, Me

FLORIANÓPOLIS, 2020.

Jeremias, Hermeson
ESTUDO DE UM MEDIDOR DE CORRENTE CONECTADO COM UMA PLATAFORMA IOT / Hermeson Jeremias ; orientação de Samir Bonho. – Florianópolis, SC, 2020.
46 p.

Trabalho de Conclusão de Curso (TCC) – Instituto Federal de Santa Catarina, Câmpus Florianópolis. CST em Sistemas Eletrônicos. Departamento Acadêmico de Eletrônica.
Inclui Referências.

1. Medidor inteligente. 2. Redes inteligentes. 3. Serviços em nuvem. I. Bonho, Samir. II. Instituto Federal de Santa Catarina. Departamento Acadêmico de Eletrônica. III. Título.

SMART METER COM INTERFACE WEB INTEGRADO A CLOUD COMPUTING

HERMESON JEREMIAS

Este trabalho foi julgado adequado para obtenção do título de Tecnólogo em Eletrônica Industrial e aprovado na sua forma final pela banca examinadora do Curso Superior de Tecnologia em Eletrônica Industrial do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 14 de Outubro, 2020.

Banca Examinadora:

_____
Samir Bonho, Me

_____
Reginaldo Steinbach, Me

_____
Everton Luiz Ferret dos Santos, Me

AGRADECIMENTOS

Agradeço, primeiramente aos meus pais, que me incentivaram todo o tempo em que estive na faculdade.

Aos professores do departamento, pelo aprendizado nestes anos de estudo.

Ao meu orientador Samir Bonho, pelo suporte e ensinamentos nesta caminhada.

E principalmente a minha noiva Rafaela, que esteve ao meu lado nos momentos mais difíceis, me apoiando e dando forças para continuar.

RESUMO

A utilização de redes inteligentes traz uma nova perspectiva para a melhoria dos serviços de distribuição de energia elétrica. Utilizando uma infraestrutura de telecomunicações que garante a conectividade de medidores de energia elétrica à Internet, é possível alimentar o sistema que controla a rede inteligente, fazendo com que o seu gerenciamento seja mais efetivo. Estes medidores, chamados de *smart meters*, podem medir variações de tensão, aumento de carga e falhas, além da energia consumida. Neste trabalho, será apresentada uma prova de conceito de um medidor inteligente. O hardware contém um sensor de corrente conectado à módulo composto por um micro controlador e um transceptor WiFi. Os dados de corrente são lidos e enviados a um serviço web alocado numa plataforma em nuvem. Informações sobre consumo de energia e fatura são gerados e podem ser visualizados por uma interface web que roda diretamente no medidor.

Palavras-chave: Medidor Inteligente. Redes inteligentes. Serviços em nuvem.

ABSTRACT

The smart grids are improving the electric power distribution by monitoring power consumption and electrical failures. In order to assure safe and reliable service, data need to be sent to the power system operators. This can be done by using smart meters, an equipment that can read voltage, current and power consumption and transmit it to the Internet. This work presents a hardware proof of concept of a smart meter. The system comprises a current sensor and a wireless microcontroller with a WiFi transceiver. Once data is read, it is sent to a cloud platform to be processed by a web server. Then the domestic user can visualize information about power consumption accessing a web page hosted directly in the smart meter.

Keywords: Smart meter. Smart grids. Cloud services.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Um conceito que está cada vez mais perto do consumidor de energia elétrica residencial é o das *Smart Grids*: “As redes inteligentes de energia, ou do inglês *Smart Grid*, são uma nova arquitetura de distribuição de energia elétrica, mais segura e inteligente, que integra e possibilita ações a todos os usuários a ela conectados. ” (CEMIG,2019).

Para uma rede inteligente funcionar é necessário um equipamento chamado de *Smart Meter*. Este é a junção do wattímetro, um equipamento utilizado para medir o consumo energético de uma estrutura, com o conceito de IoT (do inglês *Internet of Things* ou Internet das coisas, em Português), que se refere a criar uma conexão entre objetos por meio da Internet. Esse equipamento pode fornecer informações sobre o consumo em tempo real e avisar sobre falhas na rede de distribuição.

Estas informações sobre a rede elétrica de distribuição permitem que haja um gerenciamento e controle de energias de modo a melhorar o serviço das operadoras de energia elétrica. Assim como em outros países, o Brasil também está trabalhando para a implementação desse sistema. Um exemplo disso é o Projeto de Lei do Senado n° 356, de 2017 do Senador Eduardo Braga (MDB-AM) (Senado Notícias, 2018), que visa incentivar a modernização da rede de distribuição de energia no país; outro exemplo é o projeto “Redes Inteligentes Brasil”, da Agência Nacional de Energia Elétrica (ANEEL, 2010), que teve seu início em 2010 e implantou projetos pilotos em cidades como Barueri, contemplando 84.000 consumidores, e em Curitiba, contemplando 10.000 consumidores; a ideia é criar um modelo de implantação replicável para utilizar em todo o território nacional. São várias as vantagens, entre elas, podem-se citar a capacidade para resposta à demanda e controle de demanda de modo automático e a detecção de falhas na distribuição de energia.

Este trabalho pretende apresentar uma prova de conceito de um *Smart Meter,* projetado a partir de um módulo sem fio micro controlado, que irá se conectar a um servidor para recepção de dados para verificar em tempo real o consumo de energia elétrica diário e mensal de um determinado equipamento que esteja sendo monitorado. A informação salva no servidor então será carregada em uma página *Web* para que o usuário possa verificar as medições.

1.1 **Justificativa**

As falhas na distribuição de energia elétrica são muito comuns nas cidades brasileiras. Em Florianópolis, é senso comum que durante a temporada de verão as oscilações da tensão da rede, devido ao aumento de carga, e as interrupções de energia, devido às tempestades de verão, ocorrem com bastante frequência.

O conhecimento da diminuição da tensão ou a falta de energia pode deixar mais eficiente os serviços das distribuidoras. Com a infraestrutura de telecomunicações atual que permite a conectividade através da internet, a informação de status da rede pode ser transmitida entre uma unidade consumidora residencial e a distribuidora de eletricidade.

Pensando nisso será realizado um protótipo para o monitoramento do consumo energético de um equipamento, que posteriormente será armazenado em um servidor na nuvem e mostrado em uma página web. O resultado será um protótipo que poderá ser utilizado tanto por usuários domésticos que querem apenas ter a informação de seu consumo, quanto para uma distribuidora de energia elétrica que poderá utilizá-lo para verificar falhas na distribuição.

1.2 **Definição do Problema**

O monitoramento em tempo real do consumo energético de residências pode trazer uma melhora no serviço prestado pelas distribuidoras de eletricidade. Para verificação do consumo, basta a medição de corrente e tensão por um circuito elétrico. A problemática na implementação de uma rede inteligente, que soma medição e transmissão da informação, se concentra nas infraestruturas de comunicação digital disponíveis. Com o avanço da tecnologia, está cada vez mais comum o conceito de *IoT* em que vários equipamentos se comunicam através da rede Internet. Esta tecnologia pode ajudar o usuário doméstico a monitorar seu consumo energético?

1.3 **Objetivo Geral**

Desenvolver um sistema para monitorar a corrente de um equipamento.

1.4 **Objetivos Específicos**

Para chegar ao objetivo geral, foram traçados os seguintes objetivos específicos:

a) Analisar serviços de armazenamento na nuvem;

b) Criar um firmware para envio dos dados sobre o consumo energético;

c) Conceber scripts para manipular os dados recebidos pelo serviço na nuvem;

d) Gerar uma página *web* para visualização dos dados.

2 REVISÃO DA LITERATURA

Para melhor entendimento sobre o trabalho realizado será feito uma breve explicação sobre conceitos importantes para a realização do mesmo.

2.1 *Smart meter*

O termo *Smart meter* ou Medidor inteligente, em português, se refere a um equipamento eletrônico utilizado para registrar medições em tempo real de grandezas como corrente elétrica, fator de potência, consumo de água ou gás. Estas informações então são enviadas a um servidor, normalmente através de rede Wi-Fi para serem analisadas. (Smart Energy GB, 2020).

Este tipo de equipamento já está presente no mercado e suas especificações variam de fabricante para fabricante. A Tabela 1 nos mostra um breve comparativo entre produtos já inseridos no mercado e o protótipo que será construído.

Tabela 1 – Comparativo de mercado

Comparativo de mercado							
	Marca	**Modelo**	**Comunicação**	**Corrente**	**Tensão**	**Potência**	**Conexão**
1	Fronius	US-240	Wifi	4A	220V	220W	Serie
2	Prototipo	ACS712-5A	Wifi	5A	110V~220V	1100W	Serie
3	Sonoff	POW-R2	Wifi	15 A	110V ~220V	3500W	Serie
4	Tomzn Hiking	DDS238-4 W	Wifi	60A	110V~220V	13KW	Serie

Fonte: Elaboração Própria (2020).

Conforme nota-se na Tabela 1, a utilização de conexão Wi-Fi é comum no mercado em virtude da maior probabilidade de infraestrutura existente. Outro ponto importante a se observar é a diferença entre a corrente suportada pelos medidores. No mercado temos desde modelos que suportam grandes correntes, como o modelo DDS238-4 W da marca Tomzn Hiking, quanto modelos que suportam baixa corrente, como o modelo US-240 da marca Fronius. Para o protótipo foi escolhido um medidor que suporta 5A pois o foco do projeto é testar uma prova de conceito.

2.2 **Cloud Computing**

A *Cloud Computing* (Computação na nuvem, em Português) em resumo é o “[...]fornecimento de serviços de computação, incluindo servidores, armazenamento, bancos de dados, rede, software, análise e inteligência, pela Internet (“a nuvem”) para oferecer inovações mais rápidas, recursos flexíveis e econômicas de escala. ”

(Microsoft,2020), ou seja, é um serviço externo hospedado em servidores que pode ser acessado em qualquer ponto do mundo basta ter acesso à Internet.

Entre suas principais vantagens podemos citar:

a) Como o processamento de dados fica a cargo do servidor, o usuário não precisa de grandes investimentos em Hardware;

b) Facilidade no acesso e compartilhamento de arquivos, já que todos ficam no mesmo servidor;

c) Dados podem ser acessados em qualquer lugar;

d) Os custos com manutenção de servidores ficam por responsabilidade do provedor.

Há várias empresas com seus serviços de *Cloud Computing* entre elas podemos citar:

a) Azure da Microsoft;

b) AWS (do inglês Amazon Web Services) da Amazon;

c) Google Cloud Plataform da Google;

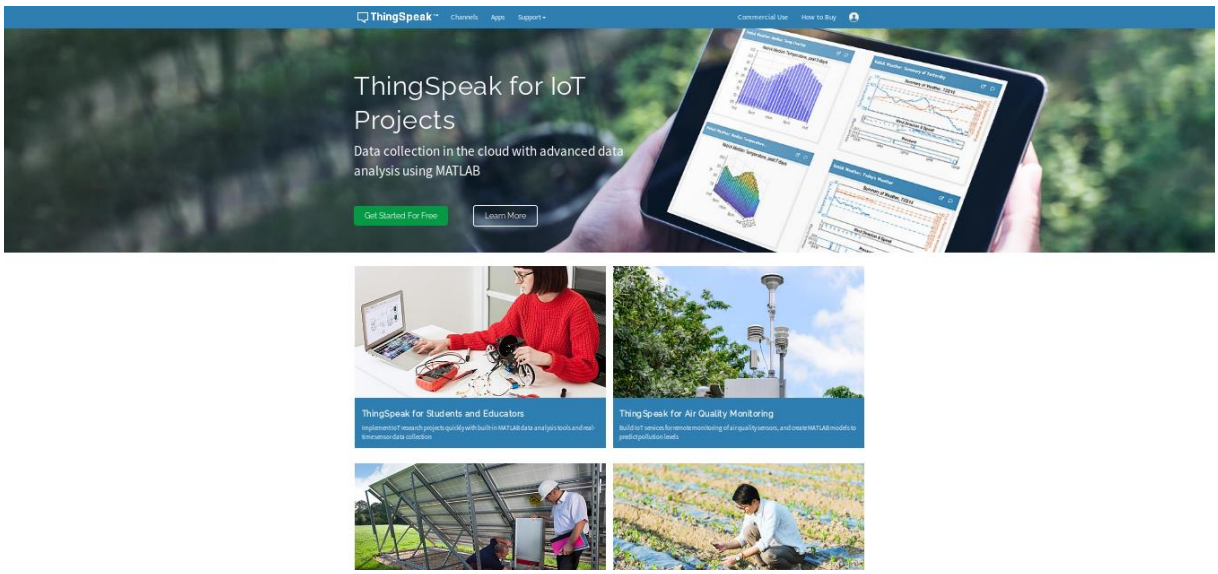
d) ThingSpeak da MathWorks.

A escolha de um ou outro serviço vai depender da aplicação, nível de segurança exigido, preço, quantidade total de dados transmitidos e capacidade de conexões simultâneas, além de outros requisitos.

2.2.1 ThingSpeak

O ThingSpeak é uma plataforma voltada para *IoT* integrada com o Matlab. Ela possui ferramentas como visualização de gráficos e triggers para automação de tarefas. Recebe upload de dados através de requisições HTTP/HTTPS contendo informações que serão gravadas, localização do canal onde serão gravadas e chave de autenticação de escrita. (S. Lemos et al., 2016). A Figura 1 mostra a página inicial do ThingSpeak.

Figura 1 – Pagina ThingSpeak



Fonte: ThingSpeak (28/09/2020). https://thingspeak.com

Entre as tantas opções presentes na Internet, escolheu-se o ThingSpeak por sua familiaridade, pois utiliza a mesma sintaxe do programa MATLAB, e pela vantagem de geração de gráficos para visualização dos dados e eventuais resultados de processamento.

2.3 **Efeito Hall**

O efeito Hall é basicamente um campo elétrico que surge em um material condutor, quando o mesmo é percorrido por uma corrente elétrica e submetido a um campo magnético.

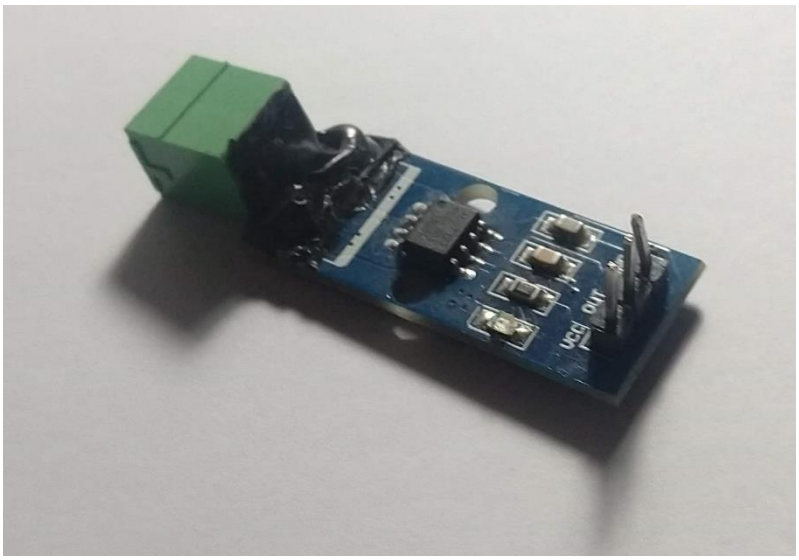
> Durante seus estudos de doutorado, Edwin Hall buscava entender qual a influência de um campo magnético externo sob um fio condutor. Ele queria entender se a força devido a este campo externo atuaria sobre os portadores de corrente elétrica ou sobre o fio como um todo. Hall acreditava que essa força magnética atuaria sobre os portadores de carga fazendo com que a corrente se deslocasse para uma determinada região do fio, e portanto, a resistência do fio iria aumentar. Apesar de não observar tal aumento na resistência do fio em seus experimentos, Hall sabia que de alguma forma a corrente elétrica era alterada sem que a resistência fosse modificada. Ele propôs a presença de um estado de stress em uma determinada região do condutor, devido ao acúmulo de portadores de carga, que originaria uma diferença de potencial transversal mais tarde conhecida como **tensão de Hall**. (Wikipedia, 2019).

2.3.1 Sensor de efeito Hall

Para realizar as medições foi utilizado o sensor de corrente

ACS712ELCTR-05B-T do fabricante Allegro™. Este fabricante fornece Sensores de efeito Hall em versões de 5A, 20A ou 30A, foi escolhido o modelo de 5A pois este valor de corrente serve como prova de conceito, além de apresentar uma maior tensão de saída, cerca de 185mV/A contra 100mV/A da versão para 20A e 66mV/A da versão para 30A, tornando assim a medição mais precisa. Além disso outra vantagem deste sensor é que ele possui isolamento interno, impedindo que o circuito ligado à rede elétrica afete o resto do circuito. A Figura 2 mostra o sensor utilizado.

Figura 2 – Sensor ACS712

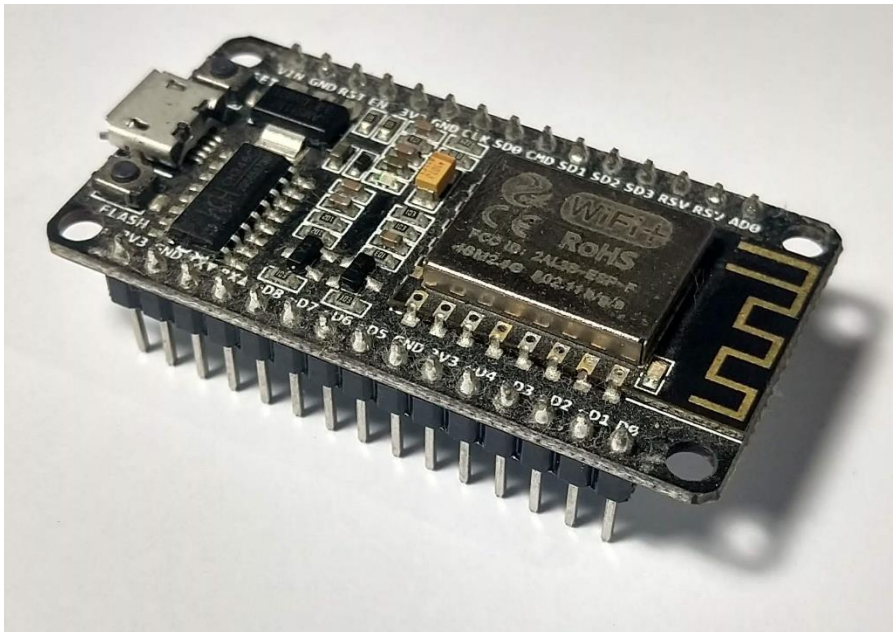


Fonte: Elaboração Própria (2020).

2.4 **Módulo NodeMcu ESP-12**

Para este protótipo foi escolhido o módulo NodeMCU ESP-12 pois esta placa possui protocolos TCP/IP integrados, garantindo acesso a redes Wifi, interface usb-serial e sua programação pode ser feita através da IDE do Arduíno tornando sua programação de fácil acesso. Quanto à conexão WiFi, o módulo opera nos padrões IEEE 802.11 b/g/n, com potência máxima de saída de 20 dBm e antena microstrip integrada. O link da camada de enlace pode ser criptografado com WPA/WPA2. Além disso, esta placa possui vantagens que facilitam o desenvolvimento de protótipos como barramento de pinos para uso em matriz de contato e regulador de tensão (tensão de saída 3.3V) para alimentação de periféricos. Sua porta *ADC* (do inglês *analogue-to-digital converter* ou conversor analógico-digital em português) suporta sinais de até 3.3V. Ela será utilizada neste projeto para ler o valor de corrente fornecido pelo ACS712. A Figura 3 mostra o módulo utilizado para o protótipo.

Figura 3 – NodeMCU

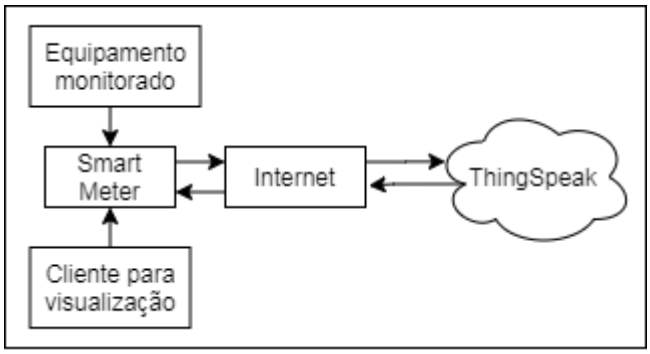


Fonte: Elaboração Própria (2020).

3 SMART METER

O sistema de medição desenvolvido possui um protótipo de leitura de corrente, que faz o papel de um *smart meter*. Este, através de conexão com a internet, envia os dados a um serviço alocado na plataforma ThingSpeak. Lá os scripts de cálculo de consumo disponibilizam gráficos para análise do dispêndio de energia. A Figura 4 nos mostra como é feita a comunicação entre as partes do projeto.

Figura 4 – Esquemático de conexões do sistema como um todo



Fonte: Elaboração Própria (2019).

Através da figura acima pode-se ver que o *Smart Meter* é o ponto principal do projeto. Ele é responsável por monitorar o equipamento e realizar a comunicação entre o ThingSpeak e o cliente final. Além disso, disponibiliza uma página web para visualização dos gráficos de consumo.

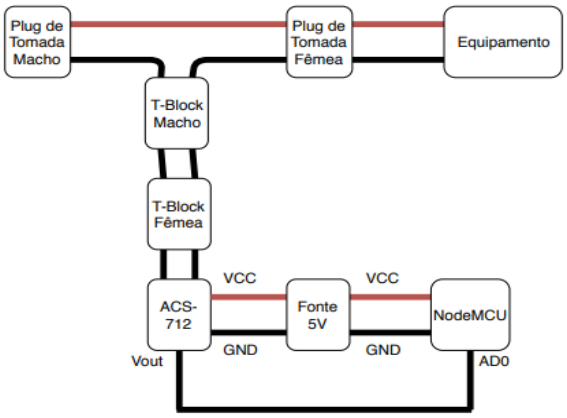
3.1 Protótipo

O protótipo elaborado é composto por hardware e firmware.

3.1.1 Hardware

O protótipo foi pensado para ter fácil manuseio e para que possa ser utilizado por qualquer usuário sem nenhum tipo de conhecimento prévio ou treinamento. A Figura 5 mostra o esquemático das ligações do hardware.

Figura 5 – Esquemático de conexões para medição de consumo



Fonte: Elaboração Própria (2019).

Na figura temos os elementos que constituem o hardware do projeto, são eles:

a) Plug de tomada Macho e Plug de Tomada Fêmea que serão utilizados para alimentar o Equipamento;

b) Equipamento que é o objeto a ser monitorado;

c) T-Block Macho ligado em série com os Plugs de Tomada e T-Block Fêmea ligado em série com o ACS-712, responsáveis por conectar o Equipamento com o sensor de corrente;

d) ACS-712, sensor utilizado para monitorar o consumo de potência do equipamento;

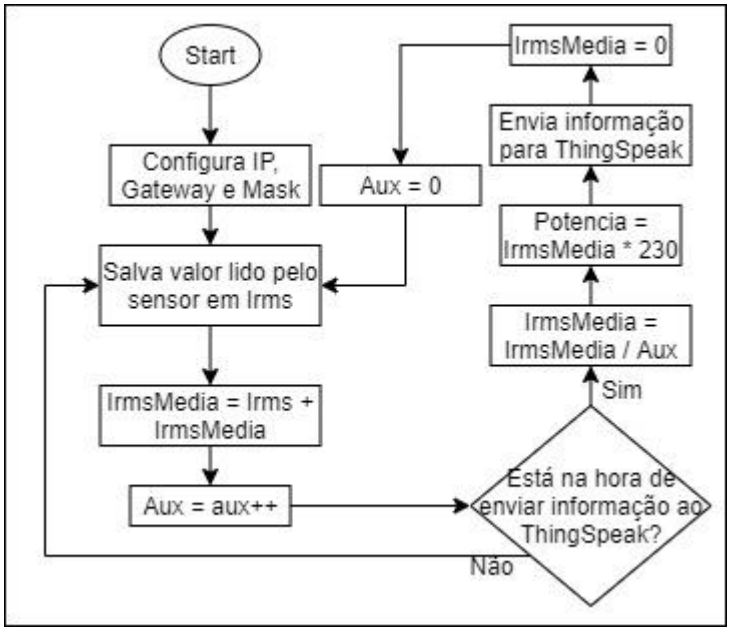
e) Fonte 5V responsável pela alimentação do ACS-712 e NodeMCU;

f) NodeMCU empregado para processar e enviar as medidas para o ThingSpeak, além de fornecer a interface *web* entre o Smart Meter e o usuário.

3.1.2 Firmware

A Figura 6 nos mostra como funciona a medição de consumo e o envio da informação para o ThingSpeak através do NodeMCU.

Figura 6 – Fluxograma para medição de consumo



Fonte: Elaboração Própria (2019).

Ao ser ligado o equipamento busca em sua memória as informações da rede WiFi que foi previamente configurada e informações de IP, máscara de rede e gateway para que possa ter acesso a conexão com o servidor na nuvem, o ThingSpeak. Após isso inicia-se a medição de consumo propriamente dita, fazendo-se uma leitura a cada segundo e incrementando-se o contador “aux” a cada medida até chegar a 15 medidas. Depois calcula-se a média da potência e envia-se as informações para o ThingSpeak. A informação enviada é calculada pela seguinte equação [2]:

$$Potencia = IrmsMedia * 220 \quad (1)$$

Onde:

Potencia = Potência média calculada

IrmsMedia = Corrente média calculada no intervalo de 15 segundos

220 = Tensão nominal da rede elétrica

3.2 **Processamento da informação**

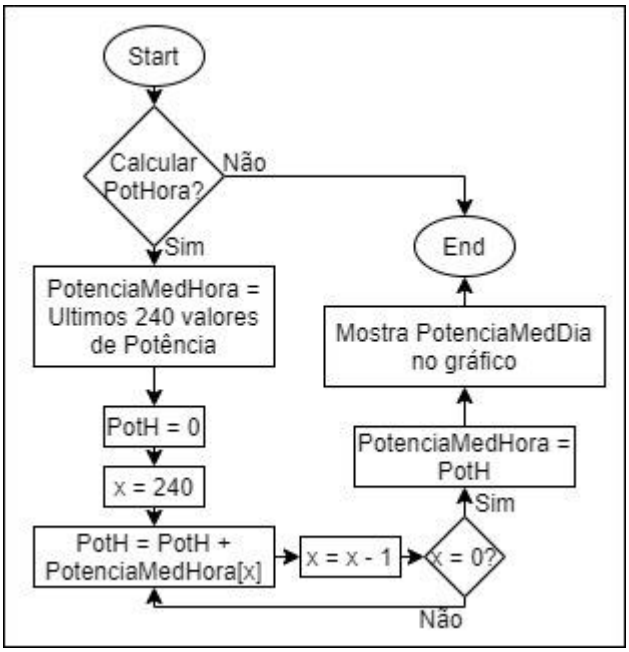
Toda informação recebida pelo ThingSpeak fica armazenada no seu servidor e pode ser acessada e processada dentro do mesmo. Para isso é usado uma

ferramenta chamada MATLAB Analysis, que consiste em scripts que serão executados em determinados momentos pré-agendados para tratamento de informações.

3.2.1 Processamento do consumo por Hora

Na Figura 7 temos o fluxograma das informações que serão mostradas ao usuário requisitar na página HTML o consumo por Hora.

Figura 7 – Fluxograma do processamento do consumo por hora



Fonte: Elaboração Própria (2019)

Primeiramente o script responsável por esta informação verifica se está na hora de calcular esta informação em caso afirmativo ele resgata os últimos 240 valores de consumo que estão no servidor e soma os valores para ter o consumo total neste período de tempo, este número de pontos são obtidos a partir da seguinte equação:

$$X = \frac{T}{A} = \frac{3600}{15} = 240 \quad (2)$$

Onde:

X = Número de pontos

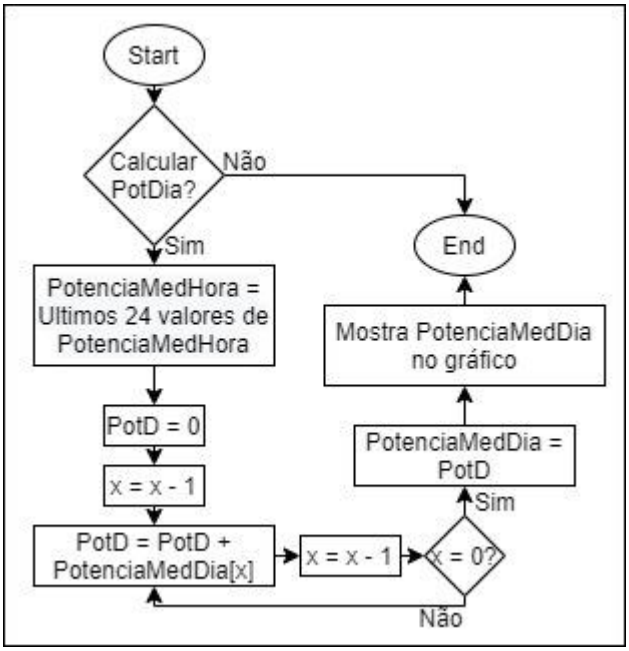
T = Tempo em segundos (1 Hora = 3600 segundos)

A = Tempo entre cada amostra (Uma amostra a cada 15 segundos)

3.2.2 Processamento do consumo por Dia

Na Figura 8 temos o fluxograma das informações que serão mostradas ao usuário requisitar na página HTML o consumo por Dia.

Figura 8 – Fluxograma do processamento do consumo por dia



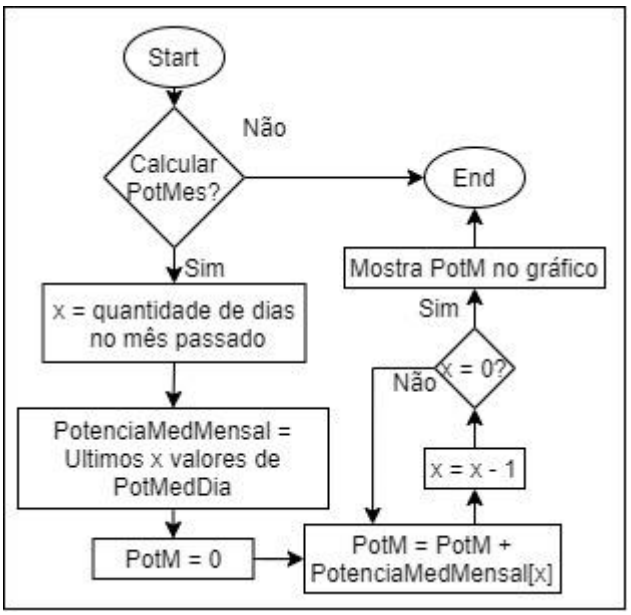
Fonte: Elaboração Própria (2019).

Primeiramente o script responsável por esta informação verifica se está na hora de calcular esta informação em caso afirmativo ele resgata os últimos 24 valores de consumo que foram calculados no script de consumo por hora, estes valores são referentes às últimas 24 Horas de consumo e soma os valores para ter o consumo total neste período de tempo.

3.2.3 Processamento do consumo por Mês

Na Figura 9 temos o fluxograma das informações que serão mostradas ao usuário requisitar na página HTML o consumo Mensal.

Figura 9 – Fluxograma do processamento do consumo por mês



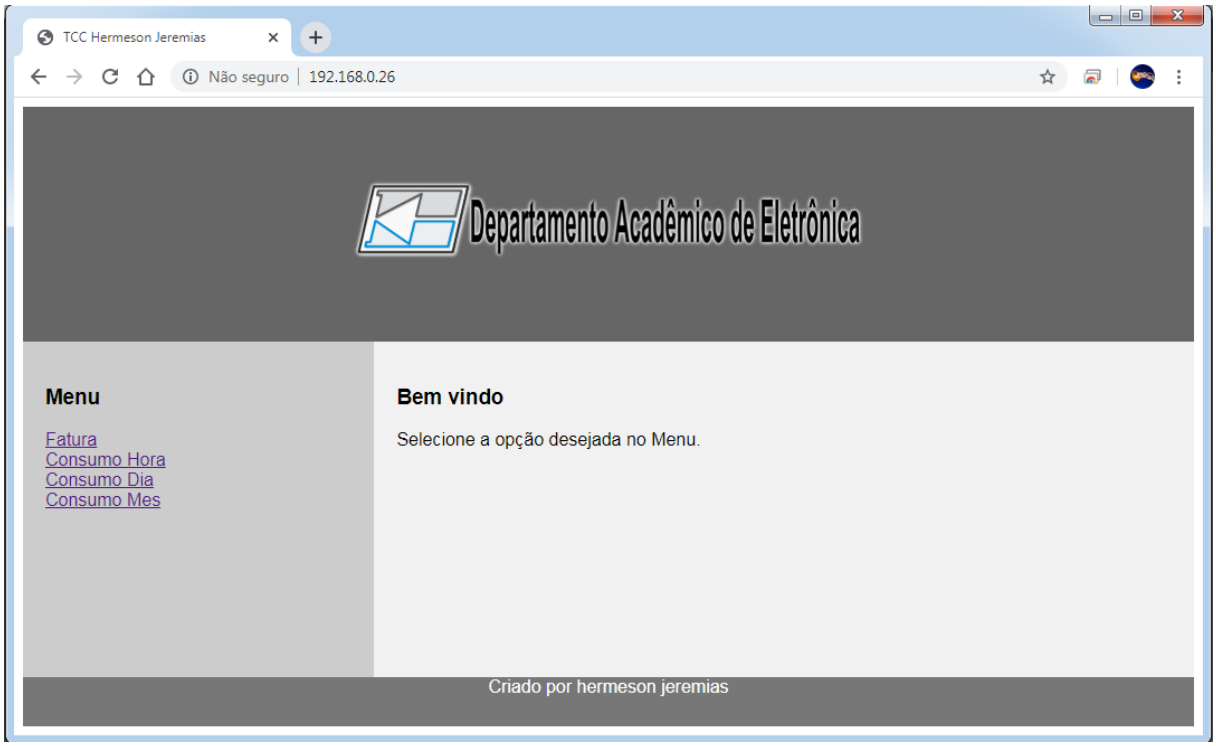
Fonte: Elaboração Própria (2019).

Para o funcionamento do script, primeiro carrega-se a informação do total de dias no mês passado e utiliza-se esta informação para verificar qual a quantidade de pontos que se irá utilizar para calcular o consumo mensal. Por exemplo, se o mês atual for Maio, ele vai carregar na variável x o número 30 que corresponde ao total de dias no mês de Abril, e irá utilizá-la para carregar e somar os últimos 30 pontos salvos na variável PotenciaMedDia.

3.3 **Interface com o usuário**

Após ser processada toda a informação é hora de mostrá-la para o usuário, para isso foi desenvolvido uma interface em HTML para ser acessada a partir do endereço IP do protótipo. A Figura 10 mostra a página que será mostrada para o usuário.

Figura 10 – Interface com o usuário



Fonte: Elaboração Própria (2020).

Nesta interface o usuário poderá escolher qual informação quer ter acesso, são elas:

a) Fatura para verificar o valor que será pago pelo consumo;

b) Consumo Hora para observar o consumo nas últimas horas;

c) Consumo Dia para ter acesso ao consumo dos últimos dias;

d) Consumo Mês para obter a potência consumida no último mês.

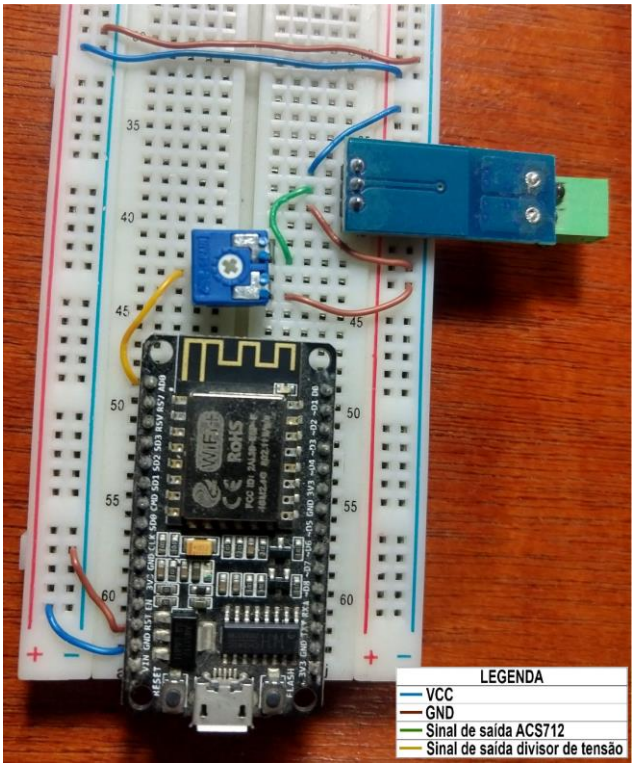
4 RESULTADOS

Neste capítulo serão abordados os resultados obtidos.

4.1 Hardware

O protótipo foi montado em uma matriz de contatos utilizando o NodeMCU, ACS712 e Trimpot, a Figura 11 mostra o protótipo.

Figura 11 – Protótipo

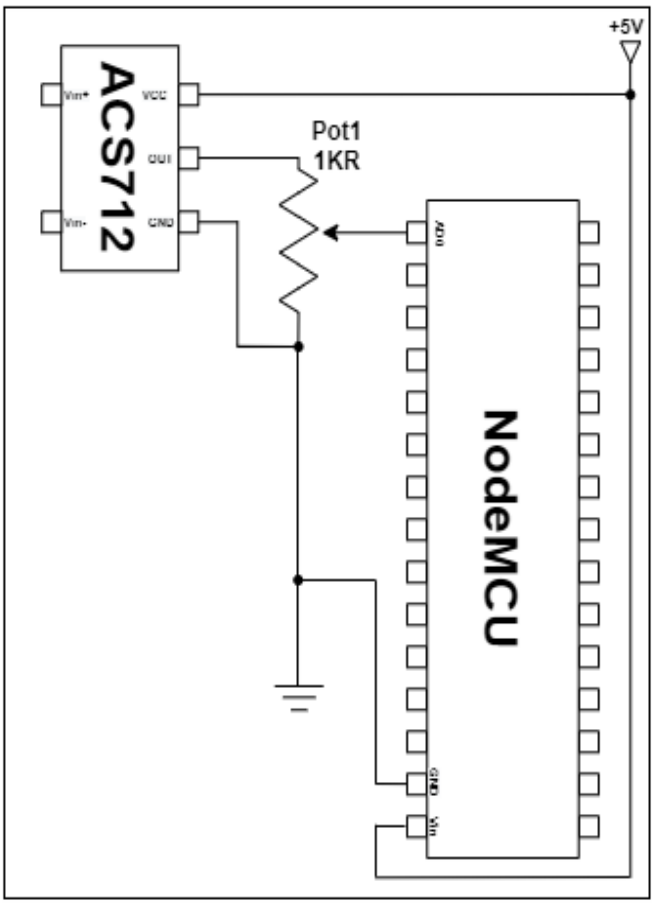


Fonte: Elaboração Própria (2020).

4.1.1 Calibração com divisor de tensão

Como o sinal de saída do leitor ACS712 varia entre 0V e 5V e a porta AD0 do NodeMCU suporta apenas sinais entre 0V e 3.3V, se fez necessário utilizar um divisor de tensão para diminuir o valor da tensão de leitura. Este divisor de tensão consiste em um Trimpot 1KR que foi ligado em 5V e ajustado para que sua saída de tenha 3.3V, ele então foi ligado ao circuito. A Figura 12 corresponde ao esquemático das ligações entre o leitor e o NodeMCU.

Figura 12 – Esquemático do Protótipo

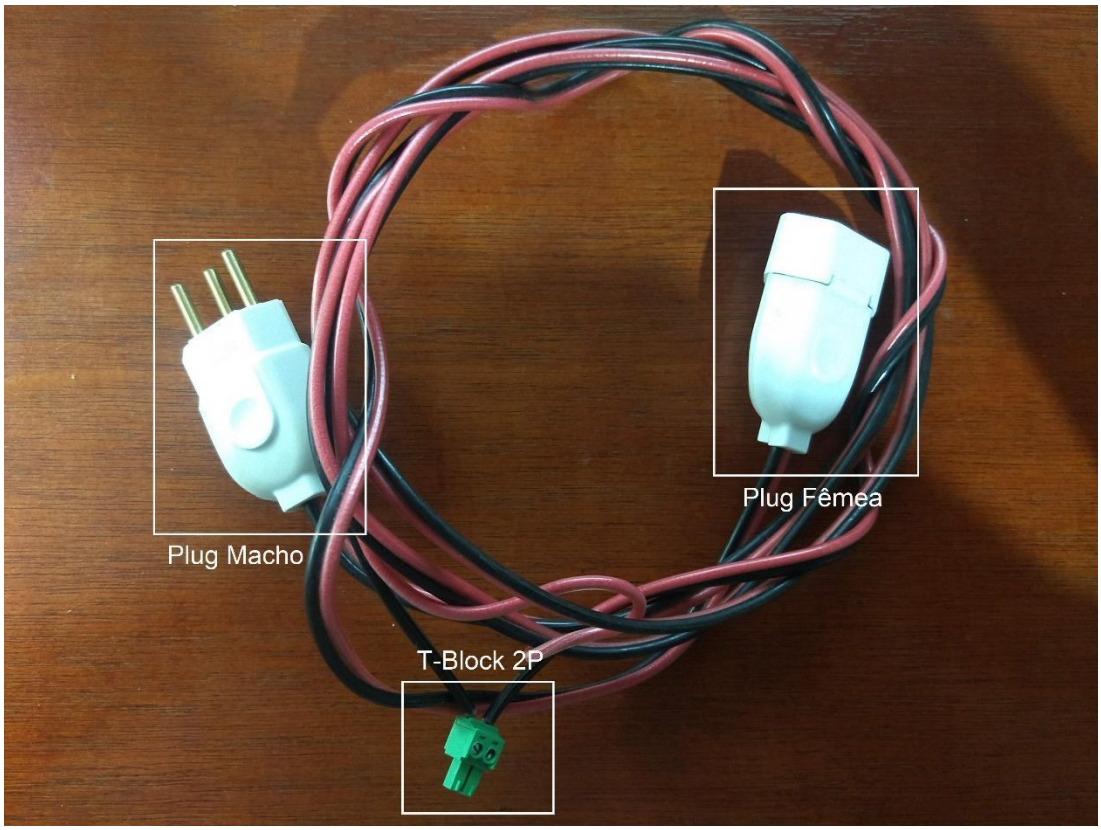


Fonte: Elaboração Própria (2020).

4.1.2 Leitura

A leitura é feita através de um cabo de força em que no meio dele foi posto um T-Block de duas posições, este plug é utilizado para se conectar no leitor de corrente para retirar amostras de leitura. A Figura 13 nos mostra melhor este cabo.

Figura 13 – Cabo para amostragem



Fonte: Elaboração Própria (2020).

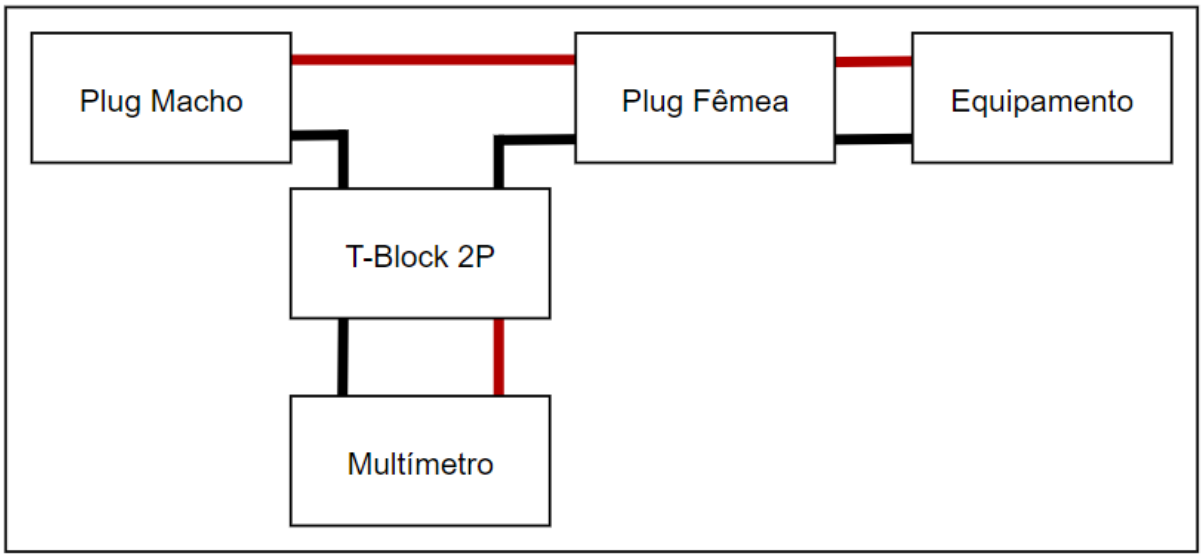
Para garantir que a leitura seja confiável, se utilizou um multímetro para conferência de dados, o modelo utilizado foi o MD-1500 da marca ICEL. Os dados obtidos pelo multímetro foram utilizados para comparar com os lidos pela porta AD0 do NodeMCU, a fim de diminuir o erro obtido com as leituras. A Figura 14 mostra os dados obtidos através da leitura com o multímetro.

Figura 14 – Leitura com multímetro

Fonte: Elaboração Própria (2020).

Para melhor entendimento de como foi efetuado a leitura com o multímetro, a Figura 15 mostra o diagrama de blocos para a leitura com multímetro.

Figura 15 – Diagrama de blocos para leitura com multímetro



Fonte: Elaboração Própria (2020).

Por fim foi efetuado as leituras através do protótipo para que possam ser comparados os resultados do protótipo com os valores obtidos com a ajuda do multímetro. A Figura 16 nos mostra as leituras feitas através do protótipo.

Figura 16 – Leituras do protótipo

```
Potencia = 754.17 W
Corrente = 3.29 A
Potencia = 755.68 W
Corrente = 3.45 A
Potencia = 793.33 W
Corrente = 3.46 A
Potencia = 796.63 W
Corrente = 3.30 A
Potencia = 759.72 W
Corrente = 3.46 A
Potencia = 794.68 W
- Informações enviadas ao ThingSpeak!
Corrente Enviado = 3.32 A
Potencia Enviado = 764.59 W
Corrente = 3.32 A
Potencia = 763.36 W
- Desconectado do ThingSpeak
```

Fonte: Elaboração Própria (2020).

A Figura 16 nos mostra 6 leituras realizadas, porém no item 3.1.2 é afirmado que são utilizadas 15 leituras para que seja tirada a média de consumo neste intervalo de tempo. Essa diferença se dá, pois, a Figura 16 nos mostra apenas uma parte das leituras. Quando se tira a média dos valores lidos pelo ACS712, i = 3,32 A

e se compara com o valor mostrado pelo Multímetro, i = 3,35 A se têm um valor parecido entre ambas medidas.

4.1.3 Erro de leitura

A Tabela 2 mostra os 15 valores lidos juntamente com uma estimativa de Desvio Padrão e Erro Médio.

Tabela 2 – Valores lidos

Valores lidos				
Medida	**Valor**	**Média**	**Desvio Padrão**	**Erro Médio**
1	3,28	3,326	0,069775149	0,018015866
2	3,31			
3	3,27			
4	3,29			
5	3,33			
6	3,32			
7	3,27			
8	3,28			
9	3,3			
10	3,28			
11	3,29			
12	3,45			
13	3,46			
14	3,3			
15	3,46			

Fonte: Elaboração Própria (2020).

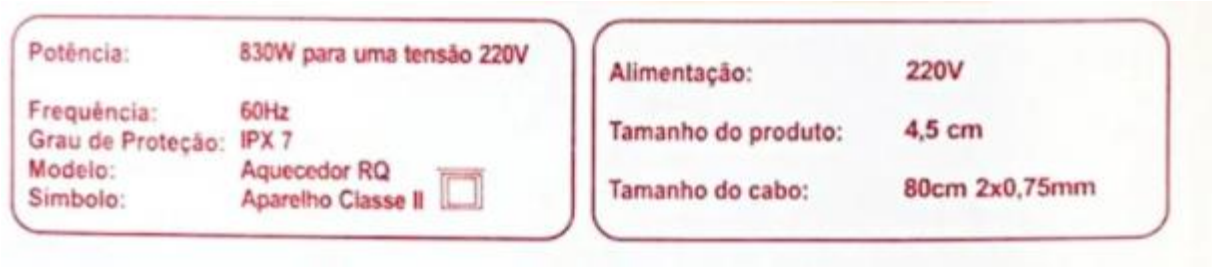
Na Tabela 2 podemos ver que os valores medidos estão próximos do esperado, tanto o valor do Desvio Padrão quanto o Erro Médio estão baixos, mostrando que os valores obtidos são satisfatórios. Como o circuito utilizado não possui tratamento de sinal, parte das discrepâncias obtidas se dão pelo ruído presente no sinal lido pelo NodeMCU, dificultando a precisão das leituras.

4.2 **Teste**

O teste foi realizado utilizando apenas uma carga puramente resistiva, pois este tipo de carga é simples de ser monitorada, facilitando os testes realizados. Para testar outros tipos de cargas seria necessário mudar o Hardware, implementando circuitos para o tratamento de sinal.

O objeto de teste foi um aquecedor de água de 830 Watts, ele foi escolhido por ser puramente resistivo e ter um consumo alto de corrente. Na Figura 17 são mostradas suas especificações.

Figura 17 – Produto monitorado



Fonte: Bwing (14 set. 2020)

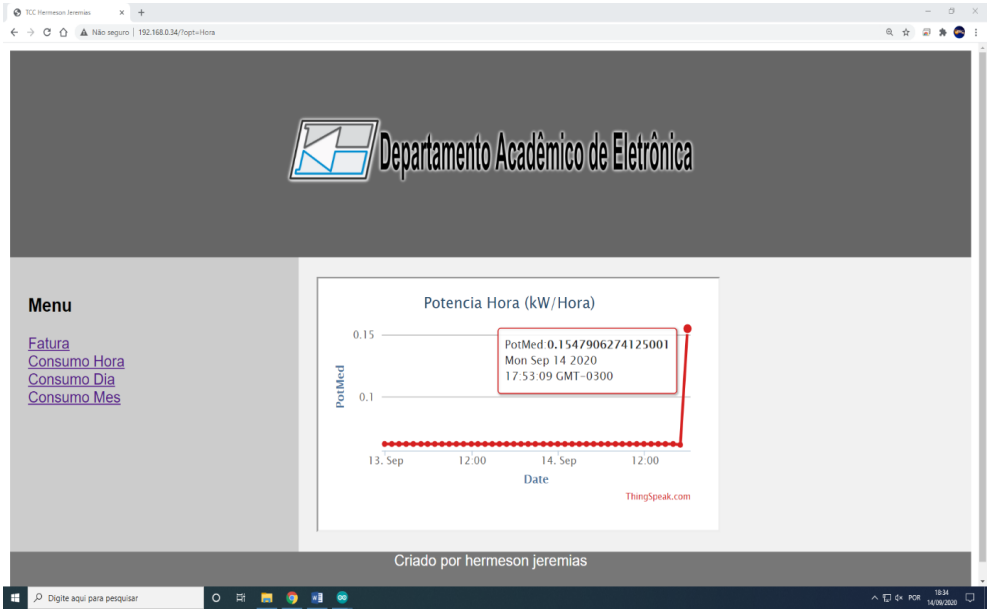
Pelas especificações do objeto monitorado sua potência é de 830W, porém o valor medido foi de 783.75W. Esta diferença se dá, pois, o valor de 830W é o máximo suportado pelo equipamento.

4.3 **Interface *Web***

Como resultado final do projeto se tem uma interface em que o usuário pode verificar os valores de potência medidos e uma estimativa do valor cobrado pela energia elétrica consumida pelo objeto monitorado. As figuras abaixo mostram os resultados observados através da interface com o usuário.

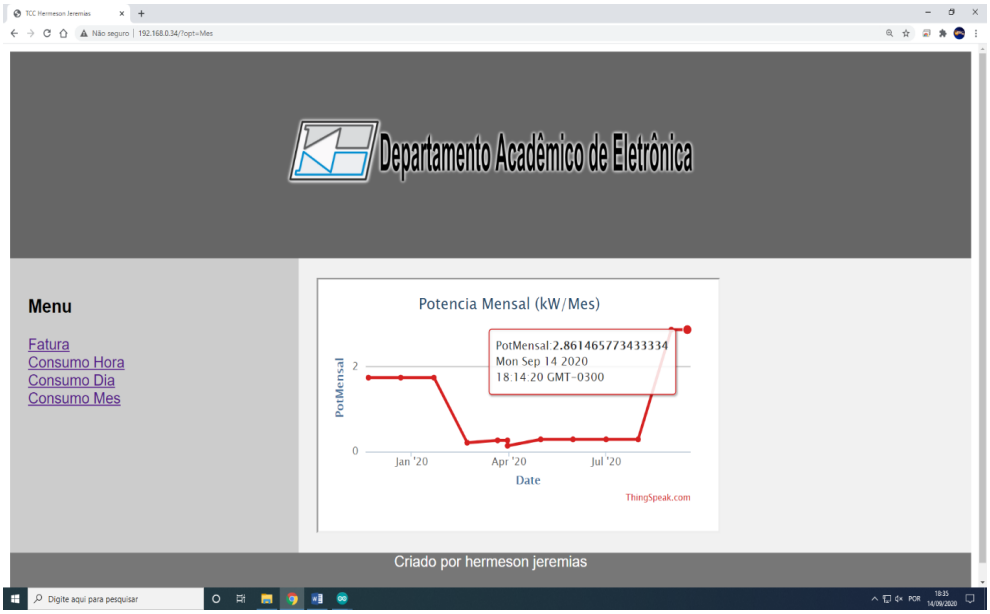
Os gráficos exibidos pelo acesso aos links do menu lateral, mostram as informações de consumo em KW/hora, kW/dia e kW/mês, além da estimativa do valor da fatura de energia elétrica que será cobrada. A Figura 18 mostra o resultado final da interface do protótipo.

Figura 18 – Resultado final

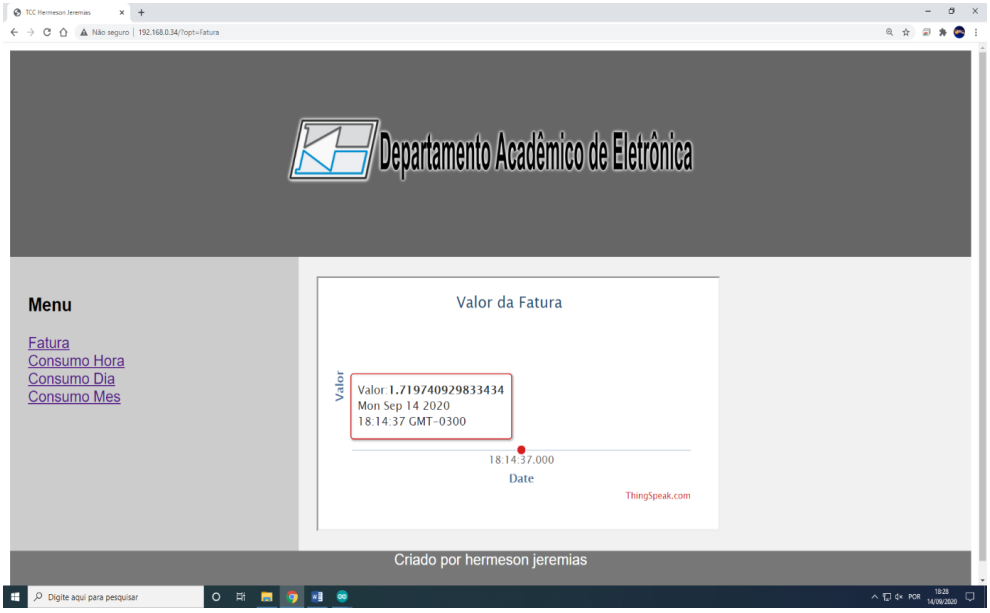


(a) Consumo por hora.

(b) Consumo por dia.



(c) Consumo por mês.



(d) Valor final do consumo mensal.

Fonte: Elaboração Própria (2020).

Conforme visto nas figuras acima, os dados são mostrados ao usuário em formato de gráficos que são lidos diretamente na interface ThingSpeak. Os valores de seus pontos podem ser lidos individualmente posicionando o ponteiro do mouse em cima do ponto desejado, facilitando assim a leitura.

5 CONCLUSÃO

As novas arquiteturas para distribuição de energia elétrica levam em consideração o conhecimento em tempo real de consumo de potência em cada unidade consumidora. Com os dados, pode-se ter um melhor gerenciamento da energia, monitorando possíveis variações de carga, de tensão e falhas. Além disso, um sistema inteligente pode mensurar as energias excedentes injetadas na rede por sistemas fotovoltaicos.

Para implementação destas redes inteligentes, se faz necessário um medidor de energia com conexão à internet, um *smart meter*: Um dispositivo com capacidade de leitura de potência e envio dos dados a um servidor em rede. Neste trabalho se desenvolveu um protótipo para medição de consumo da energia elétrica de usuários domésticos.

As informações de potência foram enviadas a um serviço em nuvem, onde os cálculos de consumo diários e mensais foram executados. Como as informações recebidas pelo equipamento estão salvas na nuvem, elas estão razoavelmente seguras e somente podem ser vistas por outro dispositivo se tiverem acesso ao número de identificação utilizado no ThingSpeak. Como esta informação está gravada em *hardcoded* no firmware do NodeMCU, usuários externos não têm acesso direto a estas informações.

A visualização do consumo é feita através de uma página web hospedada no próprio NodeMCU. Os gráficos de consumo são obtidos através de links externos que acessam os scripts armazenados na plataforma ThingSpeak.

Se avaliarmos o projeto como um todo, o resultado final foi satisfatório, pois se conseguiu cumprir com o Objetivo Geral. Porém o teste realizado pelo protótipo foi com uma carga linear, caso seja necessário a leitura de corrente para cargas não-lineares, o Hardware pode não se comportar como o esperado. Nesse caso se faria necessário atualiza-lo, utilizando por exemplo AMPOP para isolar a carga do restante do circuito e Filtro Passa Baixa para evitar ruídos de alta frequência.

Como trabalhos futuros, pode-se vislumbrar a aplicação do sistema para grandes consumidores de energia elétrica, como indústrias. Para isso, haveria necessidade da mudança do sensor de corrente, além de adaptações no hardware de

condicionamento do sinal.

Com relação a interface com o usuário, pode-se desenvolver um menu para configuração de rede; habilitar a personalização dos gráficos mostrados, mudando por exemplo a quantidade de pontos mostrados. Ademais, na segurança dos dados, existe a possibilidade de acesso à interface ser realizado através de usuário e senha. O protocolo de envio poderia ser feito através de uma sessão TLS (Transport Layer Security), garantindo a criptografia dos dados e da sessão de comunicação.

REFERÊNCIAS

ANELL. **Redes inteligentes Brasil.** Disponível em: http://redesinteligentesbrasil.org.br/. Acesso em 25 nov. 2019.

BWING. **Aquecedor para água.** Disponível em: https://www.bwimg.com.br/vaiumchima/produtos/aquecedor-para-agua---rabo-quente-1592534117.3582.jpg. Acesso em: 14 set. 2020.

CEMIG. **O QUE SÃO AS REDES INTELIGENTES DE ENERGIA?.** Disponível em: https://www.cemig.com.br/pt-br/A_Cemig_e_o_Futuro/sustentabilidade/nossos_programas/Redes_Inteligentes/Paginas/as_redes_inteligentes.aspx . Acesso em 28 nov. 2019.

MICROSOFT. **O que é computação em nuvem?**. Disponível em: https://azure.microsoft.com/pt-br/overview/what-is-cloud-computing/. Acesso em: 10 set. 2020.

S. LEMOS et al. **Estudo Comparativo de Middlewares para Internet das Coisas usando Plataformas Livres no Monitoramento de Ambientes.** Disponível em: http://sbrt.org.br/sbrt2016/anais/IC02/1570281029.pdf. Acesso em: 20 ago. 2020

SENADO NOTÍCIAS. **Projeto aprovado na CI incentiva implantação de redes elétricas inteligentes**, Jul. 2018**.** Disponível em: https://www12.senado.leg.br/noticias/materias/2018/06/12/projeto-aprovado-na-ci-incentiva-implantacao-de-redes-eletricas-inteligentes. Acesso em: 25 nov. 2019.

SMART ENERGY GB. **SMART METERS EXPLAINED.** Disponível em: https://www.smartenergygb.org/en/about-smart-meters. Acesso em 18 det. 2020.

THINGSPEAK. **THINGSPEAK.** Disponível em: https://thingspeak.com/. Acesso em 28 set. 2020.

WIKIPEDIA. **Efeito Hall**. Disponível em: https://pt.wikipedia.org/wiki/Efeito_Hall. Acesso em: 16 ago. 2020.

APÊNDICES

APÊNDICE A – Firmware embarcado no NodeMCU

```
#include <ESP8266WiFi.h>
#include <WiFiServer.h>

#include "EmonLib.h"
EnergyMonitor emon1;

//defines
#define INTERVALO_ENVIO_THINGSPEAK  15000 //intervalo entre envios de dados
ao ThingSpeak (em ms)

//constantes e variáveis globais
char EnderecoAPIThingSpeak[] = "api.thingspeak.com";
String ChaveEscritaThingSpeak = "UPMH6WLFWWV9NOUF";
long lastConnectionTime;
WiFiClient client;
WiFiClient cliente;
WiFiServer server(80);
String html;//String que armazena o corpo do site.
void EnviaInformacoesThingspeak(String StringDados);
void FazConexaoWiFi(void);
float Irms;
float IrmsMedia;
float potencia;
char ValorCorrente[50];
int aux = 0;
int opt = 0;

//Função: envia informações ao ThingSpeak
//Parâmetros: String com a informação a ser enviada
//Retorno: nenhum
void EnviaInformacoesThingspeak(String StringDados)
{
  if (client.connect(EnderecoAPIThingSpeak, 80))
  {
    //faz a requisição HTTP ao ThingSpeak
    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: " + ChaveEscritaThingSpeak + "\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(StringDados.length());
    client.print("\n\n");
    client.print(StringDados);

    lastConnectionTime = millis();
    Serial.println("- Informações enviadas ao ThingSpeak!");
  }
}

void setup()
{
  pinMode(A0, INPUT);
  emon1.current(A0, 14.4);
```

```
  Serial.begin(115200);
  WiFi.begin("Jeremias", "hermesonjeremias" );
  Serial.println("");
  Serial.print("Conectando");
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }

  IPAddress ip(192, 168, 0, 26);
  IPAddress gateway(192, 168, 0, 1);
  IPAddress subnet(255, 255, 255, 0);
  Serial.print("Configurando IP fixo para : ");
  Serial.println(ip);
  Serial.println("");
  Serial.println("Connectado");
  Serial.print ("IP: ");
  Serial.println(WiFi.localIP());
  server.begin();
  Serial.println("Servidor HTTP iniciado");
}

void loop()
{
  delay(200);
  http();
  Irms = emon1.calcIrms(1480);  // Calculate Irms only
  potencia = Irms * 230.0;
  aux++;
  IrmsMedia = Irms + IrmsMedia;
  Serial.print("Corrente = ");
  Serial.print(Irms);
  Serial.println(" A");
  Serial.print("Potencia = ");
  Serial.print(potencia);
  Serial.println(" W");

  if (client.connected())
  {
    client.stop();
    Serial.println("- Desconectado do ThingSpeak");
    Serial.println();
  }

  if (!client.connected() &&
      (millis() - lastConnectionTime > INTERVALO_ENVIO_THINGSPEAK))
  {
    IrmsMedia = IrmsMedia / aux;
    potencia = IrmsMedia * 230.0;
    sprintf(ValorCorrente, "field1=%f&field2=%f", IrmsMedia, potencia);
    EnviaInformacoesThingspeak(ValorCorrente);
    Serial.print("Corrente Enviado = ");
    Serial.print(IrmsMedia);
    Serial.println(" A");
    Serial.print("Potencia Enviado = ");
    Serial.print(potencia);
    Serial.println(" W");
    IrmsMedia = 0;
    potencia = 0;
    aux = 0;
```

```
  }
}

void http()
{
  cliente = server.available();

  if (cliente == true)
  {
    String req = cliente.readStringUntil('\r');
    Serial.println(req);

    if (req.indexOf("Fatura") != -1)
    {
      opt = 1;
    }
    else if (req.indexOf("Hora") != -1)
    {
      opt = 2;
    }
    else if (req.indexOf("Dia") != -1)
    {
      opt = 3;
    }
    else if (req.indexOf("Mes") != -1)
    {
      opt = 4;
    }

    html = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE
HTML>\r\n"
           "<html lang=""en"">"
           "<head>"
           "<title>TCC Hermeson Jeremias</title>"
           "<meta charset=""utf-8"">"
           "<meta name=""viewport"" content=""width=device-width, initial-
scale=1"">"
           "<style>"
           "* {"
           "  box-sizing: border-box;"
           "}"

           "body {"
           "  font-family: Arial, Helvetica, sans-serif;"
           "}"
           "header {"
           "  background-color: #666;"
           "  padding: 30px;"
           "  text-align: center;"
           "  font-size: 35px;"
           "  color: white;"
           "}"
           "nav {"
           "  float: left;"
           "  width: 30%;"
           "  height: 300px;"
           "  background: #ccc;"
           "  padding: 20px;"
           "}"
           "nav ul {"
           "  list-style-type: none;"
```

```
            "  padding: 0;"
            "}"
            "article {"
            "  float: left;"
            "  padding: 20px;"
            "  width: 70%;"
            "  background-color: #f1f1f1;"
            "  height: 300px;"
            "}"
            "section:after {"
            "  content: """";"
            "  display: table;"
            "  clear: both;"
            "}"
            "footer {"
            "  background-color: #777;"
            "  padding: 10px;"
            "  text-align: center;"
            "  color: white;"
            "}"
            "@media (max-width: 600px) {"
            "  nav, article {"
            "    width: 100%;"
            "    height: auto;"
            "  }"
            "}"
            "</style>"
            "</head>"
            "<body>"
            "<header>"
            "<p>"
            "<img
src=""http://sites.florianopolis.ifsc.edu.br/eletronica/files/2013/11/cropp
ed-DAELN_TEXTO_.png"" alt=""Trulli"" width=""460"" height=""72"">"
            "</header>"
            "<section>"
            "<form action=""/action_page.php"">"
            "  <nav>"
            "  <h1>Menu</h1>"
            "    <ul>"
            "      <li><a href=\"?opt=Fatura\">Fatura</a></li>"
            "      <li><a href=\"?opt=Hora\">Consumo Hora</a></li>"
            "      <li><a href=\"?opt=Dia\">Consumo Dia</a></li>"
            "      <li><a href=\"?opt=Mes\">Consumo Mes</a></li>"
            "    </ul>"
            "  </nav>"
            "  </form> "
            "  <article>";
    if (opt == 0)
    {
      html +=    "    <h1>Bem vindo</h1>"
                 "    <p>Selecione a opção desejada no Menu.<p>";
    }
    else if (opt == 1)
    {
      html +=  "<iframe width=""450"" height=""260"" style=""border: 1px
solid #cccccc;""
src=""https://thingspeak.com/channels/715015/charts/4?bgcolor=%23ffffff&col
or=%23d62020&dynamic=true&results=60&title=Valor+da+Fatura&type=line""></if
rame>";
    }
```

```
    else if (opt == 2)
    {
      html +=  "<iframe width=""450"" height=""260"" style=""border: 1px
solid #cccccc;""
src=""https://thingspeak.com/channels/715015/charts/1?bgcolor=%23ffffff&col
or=%23d62020&dynamic=true&results=50&title=Potencia+Hora+%28kW%2FHora%29&ty
pe=line""></iframe>";
    }
    else if (opt == 3)
    {
      html +=  "<iframe width=""450"" height=""260"" style=""border: 1px
solid #cccccc;""
src=""https://thingspeak.com/channels/715015/charts/2?bgcolor=%23ffffff&col
or=%23d62020&dynamic=true&results=500&title=Potencia+Dia+%28kW%2FDia%29&typ
e=line""></iframe>";
    }
    else if (opt == 4)
    {
      html += "<iframe width=""450"" height=""260"" style=""border: 1px
solid #cccccc;""
src=""https://thingspeak.com/channels/715015/charts/3?bgcolor=%23ffffff&col
or=%23d62020&dynamic=true&results=10000&title=Potencia+Mensal+%28kW%2FMes%2
9&type=line""></iframe>";
    }
    html +=  "  </article>"
             "</section>"

             "<footer>"
             "  <p>Criado por hermeson jeremias</p>"
             "</footer>"
             "</body>"
             "</html>";
    cliente.print(html);//Finalmente, enviamos o HTML para o cliente.
    cliente.stop();//Encerra a conexao.
  }
}
```

APÊNDICE B – Leitura do consumo por hora

```
% readChannelID = Channel ID do canal em que será lido os dados
readChannelID = 715010;
% PotFieldID = gráfico em que será lido os dados
PotFieldID = 2;
% readAPIKey = chave para leitura do canal
readAPIKey = 'IBW6SEDK0NW0N8M3';

% PotenciaMedHora = recebe os dados lidos
PotenciaMedHora =
thingSpeakRead(readChannelID,'Fields',PotFieldID,'NumPoints',240,'ReadKey',rea
dAPIKey)
%Transforma valores vazios de um vetor(valor =NaN) em 0
PotenciaMedHora(isnan(PotenciaMedHora))=0
% Calcula potencia media em kW
PotMedHora = mean(PotenciaMedHora)/1000;
display(PotMedHora,'Potencia consumida na ultima hora(kWh)');

% writeChannelID = Channel ID do canal em que será escrito os dados
writeChannelID = 715015;
% readAPIKey = chave para escrita do canal
writeAPIKey = 'OB5CG0IF388Q3QG6';

% Escreve os dados
thingSpeakWrite(writeChannelID,PotMedHora,'writekey',writeAPIKey);
```

APÊNDICE C – Leitura do consumo por dia

```
% readChannelID = Channel ID do canal em que será lido os dados
readChannelID = 715015;
% PotFieldID = gráfico em que será lido os dados
PotFieldID = 1;
% readAPIKey = chave para leitura do canal
readAPIKey = 'LGM5R9T97FIPYBL2';

% PotenciaMedDia = recebe os dados lidos
[PotenciaMedDia,time] =
thingSpeakRead(readChannelID,'Fields',PotFieldID,'NumPoints',24,'ReadKey',read
APIKey)
% Deleta valores NaN de um vetor
PotenciaMedDia(isnan(PotenciaMedDia))=[]
% Calcula a Potencia total do dia
PotMedDia = sum(PotenciaMedDia);
display(PotMedDia,'Potencia consumida no ultimo dia (kWh)');

% writeChannelID = Channel ID do canal em que será escrito os dados
writeChannelID = 715015;
% readAPIKey = chave para escrita do canal
writeAPIKey = 'OB5CG0IF388Q3QG6';

% Escreve os dados
thingSpeakWrite(writeChannelID,PotMedDia,'Fields',[2],'writekey',writeAPIKey);
```

APÊNDICE D – Leitura do consumo por mês

```
% Calculo do número de dias no mês;
% Salva em c o dia atual
c = datestr(now,'dd/mm/yyyy')

% Salva em y o ano e em m o mês atual
m = str2num(c(1,4:5))-1

% Verifica se o mês atual é Janeiro, caso seja muda o ano para o ano passado e
o mês para 12
if m ~= 0
    y = str2num(c(1,7:10))
else
y = str2num(c(1,7:10)) -1
m = 12
end
% Cria um calendário do mês passado para poder calcular quanto tempo falta
para medir o consumo
NumDias = calendar(y, m)
x=0
for a = 1:1:6
    for b = 1:1:7
        if NumDias(a,b) > 14
           if NumDias(a,b) ~=0
              x = x+1;
           end
       end
    end
end
NumDias(a,b)
Dias = x + 20

    %Verifica se chegou o dia da medição do consumo (dia 20)
    if c(1,1:2) == '20'

       % readChannelID = Channel ID do canal em que será lido os dados
       readChannelID = 715015;
       % PotFieldID = gráfico em que será lido os dados
       PotFieldID = 2;
       % readAPIKey = chave para leitura do canal
       readAPIKey = 'LGM5R9T97FIPYBL2';

       % PotenciaMedMensal = recebe os dados lidos
       PotenciaMedMensal = thingSpeakRead(readChannelID,'Fields',PotFieldID
,'NumPoints',Dias,'ReadKey',readAPIKey);
       % Deleta valores NaN de um vetor
       PotenciaMedMensal(isnan( PotenciaMedMensal))=[]
       % Calcula a Potencia total do mensal
       PotMedMensal = sum( PotenciaMedMensal)
       display(PotMedMensal,'Potencia consumida no ultimo Mes (kWh)');

       % writeChannelID = Channel ID do canal em que será escrito os dados
       writeChannelID = 715015;
       % readAPIKey = chave para escrita do canal
```

```
        writeAPIKey = 'OB5CG0IF388Q3QG6';
        % Escreve os dados
        thingSpeakWrite(writeChannelID,PotMedMensal,'Fields',[3],'writekey'
,writeAPIKey);
    End
```

APÊNDICE E – Leitura do valor da fatura

```
% readChannelID = Channel ID do canal em que será lido os dados
readChannelID = 715015;
% PotFieldID = gráfico em que será lido os dados
PotFieldID = 3;
% readAPIKey = chave para leitura do canal
readAPIKey = 'LGM5R9T97FIPYBL2';

% PotenciaMedMensal = recebe os dados lidos
[PotenciaMedMensal,time] =
thingSpeakRead(readChannelID,'Fields',PotFieldID,'NumPoints',1,'ReadKey',readA
PIKey)
% Calcula o valor cobrado com base no valor de cada bandeira.
if PotenciaMedMensal <= 150
          Valor = PotenciaMedMensal * 0.601; % Para bandeira 1 (Até 150kW)
      else
          Valor = (PotenciaMedMensal - 150)* 0.707359 + 150 * 0.601; % Para
bandeira 2 (A partir de 150kW)
end
display(Valor,'Valor da fatura (R$)');

% writeChannelID = Channel ID do canal em que será escrito os dados
writeChannelID = 715015;
% readAPIKey = chave para escrita do canal
writeAPIKey = 'OB5CG0IF388Q3QG6';

% Escreve os dados
thingSpeakWrite(writeChannelID,Valor,'Fields',[4],'writekey',writeAPIKey);
```