UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO

Eduardo Augusto Pfeifer

UMA REDE DE SENSORES SEM FIO EM MALHA PARA MONITORAMENTO ON-LINE DO CONSUMO DE ENERGIA NOS EDIFÍCIOS DA UNIVERSIDADE FEDERAL DE SANTA CATARINA: UM ESTUDO DE CASO

Araranguá

2018

Eduardo Augusto Pfeifer

UMA REDE DE SENSORES SEM FIO EM MALHA PARA MONITORAMENTO ON-LINE DO CONSUMO DE ENERGIA NOS EDIFÍCIOS DA UNIVERSIDADE FEDERAL DE SANTA CATARINA: UM ESTUDO DE CASO

Monografia de Conclusão de Curso submetida ao Curso de Engenharia de Computação para a obtenção do Grau de Bacharel em Engenharia de Computação.
Orientador: Prof. Dr. Fábio Rodrigues de La Rocha

Araranguá

2018

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Pfeifer, Eduardo Augusto
UMA REDE DE SENSORES SEM FIO EM MALHA PARA MONITORAMENTO ON-LINE DO CONSUMO DE ENERGIA NOS EDIFÍCIOS DA UNIVERSIDADE FEDERAL DE SANTA CATARINA : UM ESTUDO DE CASO / Eduardo Augusto Pfeifer ; orientador, Prof. Dr. Fábio Rodrigues de La Rocha, 2018.
129 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Graduação em Engenharia de Computação, Araranguá, 2018.

Inclui referências.

1. Engenharia de Computação. 2. Internet das Coisas. 3. Smart Meter. 4. MESH. I. La Rocha, Prof. Dr. Fábio Rodrigues de. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Computação. III. Título.

Eduardo Augusto Pfeifer

UMA REDE DE SENSORES SEM FIO EM MALHA PARA MONITORAMENTO ON-LINE DO CONSUMO DE ENERGIA NOS EDIFÍCIOS DA UNIVERSIDADE FEDERAL DE SANTA CATARINA: UM ESTUDO DE CASO

Esta Trabalho de Conclusão foi julgada aprovada para a obtenção do Título de "Bacharel em Engenharia de Computação", e aprovada em sua forma final pela Universidade Federal de Santa Catarina.

Araranguá, 26 de novembro 2018.

Eliane Pozzebon:77921313949

Assinado de forma digital por Eliane Pozzebon:77921313949
Dados: 2018.12.09 22:44:22 -02'00'

Profa. Dra. Eliane Pozzebon
Coordenador do Curso

Banca Examinadora:

Eliane Pozzebon:77921313949

Assinado de forma digital por Eliane Pozzebon:77921313949
Dados: 2018.12.09 22:45:03 -02'00'

Profa. Dra. Eliane Pozzebon

Assinado digitalmente por: Fabio Rodrigues de La Rocha:96300752020
2018.12.09 09:15:35 BRST

Prof. Dr. Fábio Rodrigues de La Rocha
Orientador

Tiago Oliveira Weber:00972727019

Digitally signed by Tiago Oliveira Weber:00972727019

Prof. Dr. Tiago Oliveira Weber

Aos meus pais e colegas, que sempre apoiaram minhas escolhas e nunca me deixaram desistir frente aos desafios.

AGRADECIMENTOS

Aos meus pais, Vanda Chiamenti Pfeifer e José Ernesto Graffitti Pfeifer, pelo apoio e motivação, fundamentais durante essa caminhada. Aos amigos que tive a oportunidade de conhecer durante esta jornada, por toda a ajuda e pelo conhecimento compartilhado. À Raissa Borges Curtarelli que me deu forças nos momentos difíceis e pela sua parceria nos momentos de foco.

Ao professor e orientador Fábio Rodrigues de La Rocha, por todo o conhecimento compartilhado. A Universidade Federal de Santa Catarina pela oportunidade, e por oferecer um ensino de excelência e qualidade. À todos que de alguma forma contribuíram, direta ou indiretamente, para conclusão deste trabalho.

Eu não quero acreditar, eu quero saber.

Carl Sagan

RESUMO

O gerenciamento dos recursos pelas Universidades Federais, exige a implementação de sistemas de monitoramento de consumo que permitam a visualização mais detalhada e transparente para que seja tomadas medidas de sustentabilidade mais eficiente. O objetivo deste trabalho é apresentar o desenvolvimento de um sistema de rede de sensores sem fio para monitoramento de energia elétrica de pontos específicos de consumo da Universidade. Além disso, analisar os dados coletados com finalidade de consciência do consumo e motivação para a redução de gastos de maneira mais eficiente. Para isso, utiliza-se um microcontrolador ESP32, o qual por meio de um programa, desenvolvido em linguagem C, possibilita a aquisição de dados com sensores de corrente, tensão e temperatura. Finalmente, utilizando as ferramentas Node-RED e Blynk, é desenvolvido uma pagina WEB e um aplicativo mobile Android para registrar os dados em um banco de dados e apresentando os resultados obtidos de forma gráfica, de fácil interação com o usuário, possibilitando ao mesmo o entendimento do consumo onde os pontos são instalados.

Palavras-chave: *MESH*, Internet das Coisas, Smart Meter.

ABSTRACT

The management of resources by Federal Universities requires the implementation of monitoring systems that allow the most detailed and transparent visualization to be taken sustainability measures. The objective of this work is to present the development of a wireless sensor networ for monitoring specific electrical power points of University consumption. In addition, analyze the data collected with the purpose of awareness of the consumption and motivation for the reduction spending more efficiently. For this, a microcontroller ESP32, which, through a program developed in C language, enables the acquisition of data with current sensors, voltage and temperature. Finally, using the tools Node-RED and Blynk, a WEB page is developed and an application Android mobile to register the data in a database and presenting the results obtained in graphic form, of easy interaction with the user, making it possible to understand the consumption where the dots are installed.

Keywords: *MESH*, Internet of Things, Smart Meter.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

LISTA DE SÍMBOLOS

SUMÁRIO

1 INTRODUÇÃO

Atualmente existe uma grande preocupação com o consumo de recursos naturais em face dos impactos negativos que a exploração destes produzem na natureza. Assim, o uso racional destes recursos é de suma importância para preservar a natureza.

A energia elétrica é hoje uma *commodity* e esta é produzida de diversas formas, muitas das quais danosas ao meio ambiente e como a demanda por energia elétrica é crescente, o seu valor como *commodity* tem aumentado significativamente em todo o mundo.

Para diminuir os impactos negativos na geração de energia elétrica, muitos estudos estão sendo realizados com foco em novas formas de obtenção da energia e uso eficiente da energia produzida (AYA et al., 2012).

O investimento em maior produção de energia renovável, que por consequência teria um aumento momentâneo na disponibilidade de energia elétrica, está longe de ser a melhor solução para amenizar o problema. É importante haver uma transformação no modo de consumir energia, caso contrário o risco ambiental irá persistir. Se países em desenvolvimento (com grande quantidade populacional e que estão começando a ter acesso a bens de consumo) começassem a consumir energia com os mesmos padrões de países ricos, o atual cenário de danos à natureza e aumento do consumo ampliariam os problemas ambientais (ADELLE; PALLEMAERTS, 2009).

Portanto, a eficiência impõe a necessidade do uso racional dos recursos públicos, tornando-se necessário também o uso racional dos recursos naturais. O (TCU, 2013) do Tribunal de Contas da União destaca que gerir bem os recursos financeiros e naturais é obrigação de todo e qualquer agente público, o acórdão destaca ainda, a necessidade de um plano de ação visando orientar e incentivar todos os órgãos e entidades da Administração Pública Federal, a adotarem medidas para aumento da sustentabilidade e eficiência no uso dos recursos naturais em especial os recursos: papel, água e energia elétrica.

Uma forma de racionar o uso de energia elétrica nas Universidades Federais é detectar se a rede elétrica apresenta um consumo maior do que o esperado o que pode indicar perda de energia em algum ponto. Além disso, esse monitoramento do consumo de energia é benéfico para detectar a adequação de elementos como disjuntores, diâmetro de fios, quadros de distribuição, etc. além de revelar se o consumo da energia de um prédio é compatível com a quantidade de energia disponibilizada

na entrada do mesmo.

Recentemente tecnologias de baixo custo permitem a construção de módulos sensores capazes de capturar dados de determinado ambiente em enviá-los sem fio para um computador de destino que pode ser um computador dentro da universidade ou mesmo instalado em qualquer parte do mundo (servidor). Neste trabalho, propõe-se um sistema embarcado de baixo custo para o monitoramento elétrico dos prédios da universidade. Os dados capturados são enviados para um computador servidor onde podem ser visualizados, revelando em tempo real o perfil de consumo de energia ao longo do tempo. Além do consumo de energia (corrente) existem outras variáveis de interesse como tensão e mesmo temperatura do disjuntor que podem ser relevantes e também podem ser capturadas.

1.1 MOTIVAÇÃO E JUSTIFICATIVA

A Universidade Federal de Santa Catarina (UFSC) gasta cerca de 18 milhões com energia elétrica e consome 31.205.492 KwH ao ano, o equivalente a uma cidade de cerca de 40.000 habitantes (UFSC, 2017). De acordo com o Plano de Logística Sustentável (PLS) da UFSC (FLEESON et al., 2017), uma ferramenta de planejamento que permite aos órgãos e entidades estabelecerem práticas de sustentabilidade e racionalização de gastos e processos na Administração Pública, define que o consumo de energia deve ser reduzido em 5% por m^2 construído. Uma das principais formas de atingir essa meta é através da redução do desperdício, que segundo pesquisas, representa 40% do consumo de prédios públicos.

Sendo assim, a UFSC lançou a campanha "Reduzir o Consumo é Transformar o Mundo" com apoio do Programa UFSC Sustentável que visa incorporar a sustentabilidade nas práticas cotidianas da UFSC desde a esfera administrativa ao ensino, pesquisa e extensão, através de uma mudança de cultura organizacional. A campanha tem por objetivo fomentar a economia de recursos como energia, água, copos plásticos, materiais de expediente, entre outros (UFSC, 2013b). O primeiro eixo abordado é a redução no consumo de energia elétrica, tendo em vista os significativos custos ambientais e econômicos relacionados a energia.

Com base no propósito desse programa, a elaboração de um sistema que possa monitorar o consumo energético de vários pontos distintos tais como: elevadores, ar-condicionados, geladeiras, salas de aula e inclusive prédios inteiros, pode-se apresentar um bom meio de aquisição de informações de dados históricos para uma possível tomada de decisão em relação a conscientização e prevenção de gastos desnecessários.

1.2 OBJETIVOS

Esta seção apresenta o objetivo geral e os objetivos específicos desta monografia de conclusão de curso.

1.2.1 Objetivos Gerais

Desenvolver um sistema de monitoramento do consumo de energia elétrica de baixo custo baseado em uma rede de sensores sem fio em

topologia *mesh* com possibilidade de acesso aos dados on-line através de plataformas WEB e Mobile.

1.2.2 Objetivos Específicos

1. Realizar a leitura instantânea do consumo energético em um determinado ponto;

2. Criar uma rede de sensores sem fio com topologia *mesh* para envio dos dados coletados a partir dos sensores;

3. Disponibilizar os dados coletados para acesso on-line;

4. Utilizar ferramentas já existentes para visualização das informações a partir de gráficos e indicadores por meio de interfaces WEB e aplicativo Android.

1.3 METODOLOGIA

Este trabalho tem como propósito o desenvolvimento de um sistema de monitoramento do consumo de energia elétrica que, através de uma rede de sensores sem fio com topologia *mesh*, permite a transmissão e visualização dos dados coletados através de uma interface *WEB* e *Mobile* (Android). O *software* de controle e aquisição dos dados será desenvolvido em linguagem de programação C/C++ e fará uso das bibliotecas Emonlib (cálculo da corrente, tensão e potência) e painlessMesh (controle da rede *mesh*)

O microcontrolador utilizado para adquirir os dados e transmitir é o ESP32, em específico a placa NodeMCU-32S. Os dados transmitidos por ele será disponibilizado em um servidor em nuvem da IBM que, por meio da integração dos serviços da plataforma Node-RED, será implementado um site e a configuração de um aplicativo mobile utilizando a ferramenta Blynk, onde os gráficos (em ambas as interfaces) serão relacionados a cada dispositivo de leitura do consumo.

1.4 ORGANIZAÇÃO DO TRABALHO

Esta monografia está organizada em mais cinco capítulos que abordam os seguintes conteúdos:

O **Capítulo 2** começa com uma introdução ao termo Internet das Coisas (IoT), encaminha-se para sua perspectiva histórica e apresenta as tecnologias que abrangem a área como: Arquitetura dos dispositivos, Protocolos e Armazenamento de dados.

O **Capítulo 3** mostra situação atual em relação ao consumo de energia elétrica da Universidade Federal de Santa Catarina, apontando números de consumo e gastos financeiros.

O **Capítulo 4** tem como objetivo fazer uma breve apresentação sobre o monitoramento de consumo, focando na tecnologia conhecida como Medidores Inteligentes (*Smart Meters*).

O **Capítulo 5** mostra quais são os ambientes de desenvolvimento utilizados no projeto, ou seja, a IDE utilizada para a criação do software embarcado, quais microcontroladores, ferramentas de comunicação, de interface e bibliotecas fundamentais do trabalho.

O **Capítulo 6** apresenta o detalhamento do sistema proposto com o intuito de realizar a medição do consumo de energia. Tópicos como, requisitos dos sistema e Aquisição de dados são abordados.

No **Capítulo 7** é apresentado os testes e resultados realizados de acordo com a proposta do trabalho.

No **Capítulo 8** são apresentadas as considerações finais e propostas para trabalhos futuros.

Finalmente, o **Anexo A** contém os códigos fonte do projeto, bem como as bibliotecas utilizadas.

2 INTERNET DAS COISAS

2.1 DEFINIÇÃO DE INTERNET DAS COISAS

Internet das coisas (IoT [1]) é o um termo criado em 1999 por Kevin Ashton do MIT para descrever a ideia de conectar os itens usados do dia a dia (coisas) à rede mundial de computadores (OLSON, 2016),(ATZORI; IERA; MORABITO, 2010),(MIORANDI et al., 2012). Até então pensava-se na Internet como uma rede que conectava apenas pessoas. Hoje o cenário é tal que existem mais dispositivos eletrônicos do dia a dia acessando a rede do que pessoas. Um bom exemplo disso é a constatação que em 2011 a população da terra atingiu 7 bilhões de pessoas e o número de dispositivos na rede atingiu 13 bilhões. A expectativa para 2020 é que existirão 50 bilhões de dispositivos na rede e a expectativa é que a população mundial esteja em apenas 7.6 bilhões. A IoT não é apenas formada por dispositivos e servidores mas também por sistemas de comunicação entre estes elementos, chamada de comunicação M2M (machine-to-machine).

A União Internacional de Telecomunicações (ITU, do inglês International Telecommunication Union) define a IoT como "Uma infraestrutura global para a sociedade da informação, permitindo serviços avançados através da interconexão (física e virtual) de coisas baseadas em tecnologias interoperáveis de informação e comunicação, existentes e em evolução "(DAVIES et al., 2012).

É definida também por (YAN et al., 2008) como uma evolução radical da Internet atual em uma rede de objetos interconectados uma vez que não só colhe informações do ambiente e interage com o mundo físico, mas também usa os padrões existentes da Internet para fornecer serviços de transferência de informações, de análise, aplicações e comunicações.

A ideia básica do IoT será permitir uma conexão autônoma e segura e troca de dados entre dispositivos e aplicações do mundo real (FAN; CHEN, 2010) onde o IoT vai incorporar alguma inteligência em objetos conectados à Internet para se comunicar, trocar informações, tomar decisões, invocar ações. Tende assim fornecer serviços surpreendentes que o torna em crescente popularidade para as instituições acadêmicas, indústrias, bem como governos uma vez que tem o potencial de trazer significativos benefícios pessoais, profissionais e econômicos.

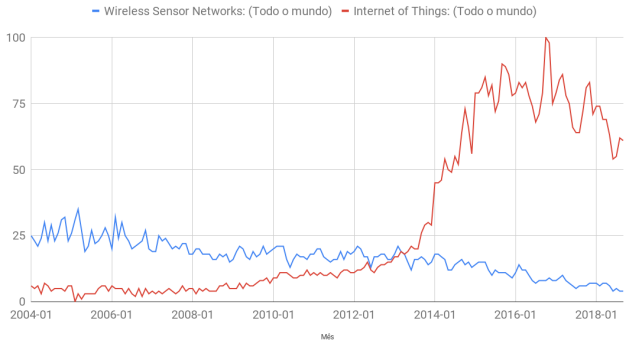
[1] *Internet of Things*

A IoT tem alterado aos poucos o conceito de redes de computadores, neste sentido, é possível notar a evolução do conceito ao longo do tempo como mostrado a seguir. Para (TANENBAUM, 2002), Rede de Computadores é um conjunto de computadores autônomos interconectados por uma única tecnologia. Entende-se que tal tecnologia de conexão pode ser de diferentes tipos (fios de cobre, fibra ótica, ondas eletromagnéticas ou outras). Em 2011, Peterson definiu em (PETERSON; DAVIE, 2011) que a principal característica das Redes de Computadores é a sua generalidade, isto é, elas são construídas sobre dispositivos de propósito geral e não são otimizadas para fins específicos tais como as redes de telefonia e TV. Já em (KUROSE; ROSS, 2012), os autores argumentam que o termo "Redes de Computadores" começa a soar um tanto envelhecido devido à grande quantidade de equipamentos e tecnologias não tradicionais que são usadas na Internet.

2.2 PERSPECTIVA HISTÓRICA DA IOT

Kevin Ashton comentou, em junho de 2009 (ASHTON, 2009), que o termo Internet of Things foi primeiro utilizando em seu trabalho intitulado "I made at Procter and Gamble" em 1999. Na época, a IoT era associada ao uso da tecnologia RFID. Contudo, o termo ainda não era foco de grande número de pesquisas como pode ser visto na Figura 1.

Figura 1 – Volume de pesquisas no Google sobre *Wireless Sensor Networks* e *Internet of Things*.

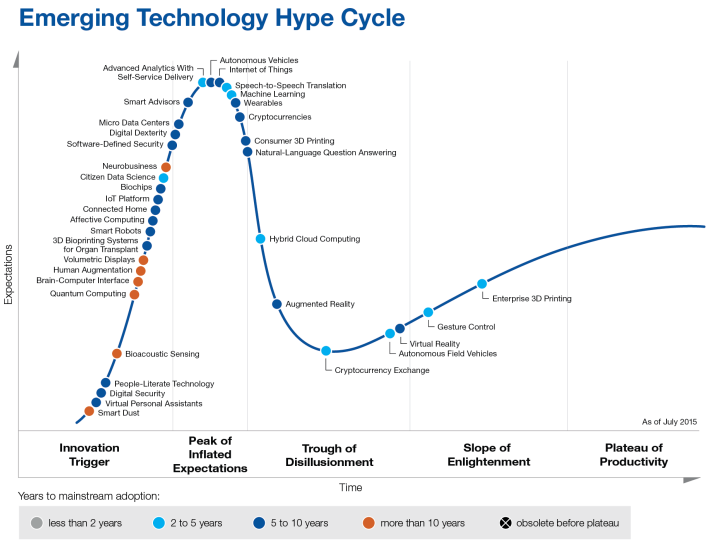


Fonte: Google Trends, 2018

Por volta de 2005, o termo bastante procurado (tanto pela academia quando indústria) e que apresenta relação com a IoT foi Redes de Sensores Sem Fio (RSSF) (do inglês *Wireless Sensor Networks* – WSN). Estas redes trazem avanços na automação residencial e industrial (T.; MUKHOPADHYAY, 2013), bem como técnicas para explorar as diferentes limitações dos dispositivos (e.g., memória e energia), escalabilidade e robustez da rede (LOUREIRO et al., 2003). Nos anos seguintes (entre 2008 e 2010), o termo Internet das Coisas ganhou popularidade rapidamente. Isto se deve ao amadurecimento das RSSFs e ao crescimento das expectativas sobre a IoT. A Figura 2 mostra que em 2010, as buscas para IoT dispararam chegando a ultrapassar as pesquisas sobre RSSFs.

A IoT foi identificada como uma tecnologia emergente em 2012 por especialistas da área (GARTNER, 2015). A Figura 2 apresenta uma maneira de representar o surgimento, adoção, maturidade e impacto de diversas tecnologias chamada de Hype Cycle, ou "Ciclo de Interesse".

Figura 2 – Tecnologias emergentes.



Fonte: (GARTNER, 2015)

O pesquisador Clarke , pesquisador do MIT, propôs a curva dos dois elefantes, que em 2012, foi previsto que a IoT levaria entre cinco e dez anos para ser adotada pelo mercado e, hoje, é vivenciado o maior

pico de expectativas sobre a tecnologia no âmbito acadêmico e industrial. Também pode-se notar o surgimento das primeiras plataformas de IoT que têm gerado uma grande expectativa de seu uso. Estes fatos certamente servem de incentivo para despertar a curiosidade do leitor para a área, bem como indica o motivo do interesse da comunidade científica e industrial para a IoT.

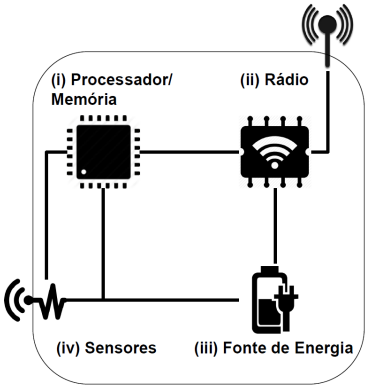
2.3 DISPOSITIVOS E TECNOLOGIAS DE COMUNICAÇÃO

Esta seção aborda em mais detalhes a arquitetura básica dos dispositivos inteligentes e as tecnologias de comunicação, principalmente as soluções de comunicação sem fio que tendem a se popularizar no ambiente de IoT.

2.3.1 Arquiteturas Básica dos Dispositivos

A arquitetura básica dos objetos inteligentes é composta por quatro unidades: processamento/memória, comunicação, energia e sensores/ atuadores (PERES et al.,). A Figura 3 apresenta uma visão geral da arquitetura de um dispositivo e a interligação entre seus componentes, os quais são descritos a seguir:

Figura 3 – Arquitetura dos dispositivos.



Fonte: (PERES et al.,)

1. **Unidade(s) de processamento/memória:** composta de uma

memória interna para armazenamento de dados e programas, um microcontrolador e um conversor analógico-digital para receber sinais dos sensores. As CPUs utilizadas nesses dispositivos são, em geral, as mesmas utilizadas em sistemas embarcados e comumente não apresentam alto poder computacional. Frequentemente existe uma memória externa do tipo flash, que serve como memória secundária, por exemplo, para manter um "log" de dados. As características desejáveis para estas unidades são consumo reduzido de energia e ocupar o menor espaço possível.

2. **Unidade(s) de comunicação:** consiste de pelo menos um canal de comunicação com ou sem fio, sendo mais comum o meio sem fio. Neste último caso, a maioria das plataformas usam rádio de baixo custo e baixa potência. Como consequência, a comunicação é de curto alcance e apresentam perdas frequentes.

3. **Fonte de energia:** responsável por fornecer energia aos componentes do objeto inteligente. Normalmente, a fonte de energia consiste de uma bateria (recarregável ou não) e um conversor AC-DC e tem a função de alimentar os componentes. Entretanto, existem outras fontes de alimentação como energia elétrica, solar e mesmo a captura de energia do ambiente através de técnicas de conversão (e.g., energia mecânica em energia elétrica), conhecidas como energy harvesting.

4. **Unidade(s) de sensor(es)/atuador(es):** realizam o monitoramento do ambiente no qual o objeto se encontra. Os sensores capturam valores de grandezas físicas como temperatura, umidade, pressão e presença. Atualmente, existem literalmente centenas de sensores diferentes que são capazes de capturar essas grandezas. Atuadores, como o nome indica, são dispositivos que produzem alguma ação, atendendo a comandos que podem ser manuais, elétricos ou mecânicos.

2.3.2 Tecnologias de Comunicação

Esta seção trata das principais tecnologias de comunicação utilizadas em IoT, identificando as características mais relevantes de cada um delas.

Ethernet. O padrão Ethernet (IEEE 802.3) foi oficializado em 1983 pelo IEEE e está presente em grande parte das redes locais com fio existentes atualmente. Sua popularidade se deve à simplicidade,

facilidade de adaptação, manutenção e custo. Atualmente, existem dois tipos de cabos: par trançado e fibra óptica, que oferecem taxas de comunicação diferentes. Os cabos de par trançado podem atingir taxas de até 1 Gbps (categoria 5), limitados a 100m (para distâncias maiores é necessário o uso de repetidores). Os cabos de fibra óptica alcançam taxas de 10 Gbps, limitados a 2000 m. O uso do padrão Ethernet é sugerido para dispositivos fixos, i.e., sem mobilidade, o que pode ser inadequado para essas aplicações.

Wi-Fi. O padrão IEEE 802.11 (*Wi-Fi*) [2] define um conjunto de padrões de transmissão e codificação. Desde o seu lançamento em 1997, já foram propostas novas versões do padrão IEEE 802.11 e, atualmente, a versão IEEE 802.11ac prevê taxas de comunicação de 600 Mbps ou 1300 Mbps. O Wi-Fi foi desenvolvido como uma alternativa ao padrão cabeado Ethernet, com pouca preocupação com dispositivos que possuem consumo energético limitado, como é o caso das aplicações para IoT. Assim, não se espera que muitos dispositivos utilizados em IoT adotem o padrão Wi-Fi como principal protocolo de comunicação. Contudo, o Wi-Fi possui algumas vantagens, como alcance de conexão e vazão, o que o torna adequado para navegação na Internet em dispositivos móveis, como smartphones e tablets. A principal desvantagem do Wi-Fi é o maior consumo de energia, quando comparado com outras tecnologias de comunicação sem fio.

ZigBee. O padrão ZigBee é baseado na especificação do protocolo IEEE 802.15.4 para a camada de enlace. As suas principais características são a baixa vazão, reduzido consumo energético e baixo custo. O ZigBee opera na frequência 2.4 GHz (faixa ISM), mas é capaz de operar em outras duas frequências, 868MHz e 915 MHz. Essa tecnologia pode alcançar uma taxa máxima de 250 kbps, mas na prática temos taxas inferiores . O ZigBee também permite que os dispositivos entrem em modo sleep por longos intervalos de tempo para economizar energia e, assim, estendendo a vida útil do dispositivo.

O padrão ZigBee é mantido pela ZigBee Alliance, organização que é responsável por gerir o protocolo e garantir a interoperabilidade entre dispositivos. O padrão ZigBee pode ser usado com o protocolo IP (incluindo o IPv6) e também utilizando a topologia em malha (Mesh) [3]. O ZigBee já é adotado em aplicações residenciais e comerciais e sua utilização em IoT depende de um gateway, dispositivo responsável por

[2]Por um abuso de linguagem, chamamos o padrão IEEE 802.11 de Wi-Fi, que na verdade representa a "Aliança Wi-Fi" (http://www.wi-fi.org/) responsável por certificar a conformidade de produtos que seguem a norma IEEE 802.11

[3]https://www.zigbee.org/zigbee-for-developers/network-specifications

permitir a comunicação entre dispositivos que usam ZigBee e os que não usam (ALLIANCE, 2002).

Bluetooth Low Energy. O Bluetooth é um protocolo de comunicação proposto pela Ericsson para substituir a comunicação serial RS-232 3. Atualmente, o Bluetooth Special Interest Group é responsável por criar, testar e manter essa tecnologia. Além disso, Bluetooth é uma das principais tecnologias de rede sem fio para PANs – Personal Area Networks, que é utilizada em smartphones, headsets, PCs e outros dispositivos. O Bluetooth se divide em dois grupos: Bluetooth clássico que por sua vez se divide em Basic Rate/Enhanced Data Rate (BR/EDR), que são as versões 2.0 ou anteriores e o Bluetooth High Speed (HS), versão 3.0; e o Bluetooth Low Energy (BLE), versão 4.0 ou superior. As versões mais antigas do Bluetooth, focadas em aumentar a taxa de comunicação, tornou o protocolo mais complexo e, por consequência, não otimizado para dispositivos com limitações energéticas. Ao contrário das versões anteriores, o BLE possui especificação voltada para baixo consumo de energia, permitindo dispositivos que usam baterias do tamanho de moedas (coin cell batteries).

Atualmente, o BLE possui três versões: 4.0, 4.1 e 4.2, lançadas em 2010, 2013 e 2014, respectivamente. BLE 4.0 e 4.1 possuem um máximo de mensagem (Maximum Transmit Unit – MTU) de 27 bytes, diferentemente da mais nova versão (BLE 4.2) que possui 251 bytes. Outra diferença entre as versões é o tipo de topologia de rede que cada versão pode criar. Na versão 4.0, apenas a topologia estrela é disponível, ou seja, cada dispositivo pode atuar exclusivamente como master ou como slave. A partir da versão 4.1, um dispositivo é capaz de atuar como master ou slave simultaneamente, permitindo a criação de topologias em malha. Recentemente foi proposta uma camada de adaptação para dispositivos BLE, similar ao padrão 6LoWPAN, chamada de 6LoBTLE. A especificação do 6LoBTLE pode ser consultada na RFC 76684 [4].

3G/4G. Os padrões de telefonia celular 3G/4G também podem ser aplicados à IoT. Projetos que precisam alcançar grandes distâncias podem aproveitar as redes de telefonia celular 3G/4G. Por outro lado, o consumo energético da tecnologia 3G/4G é alto em comparação a outras tecnologias. Porém, a sua utilização em locais afastados e baixa mobilidade podem compensar esse gasto. No Brasil, as frequências utilizadas para o 3G são 1900MHz e 2100MHz (UMTS), enquanto o padrão 4G (LTE) utiliza a frequência de 2500 MHz. A taxa de comunicação alcançada no padrão 3G é de 1 Mbps e no padrão 4G 10 Mbps

[4] https://tools.ietf.org/html/rfc7668

5.

LoRaWan. A especificação LoRaWAN (Long Range Wide Area Network) foi projetada para criar redes de longa distância, numa escala regional, nacional ou global, formada por dispositivos operados por bateria e com capacidade de comunicação sem fio. A especificação LoRaWan trata de requisitos presentes na IoT como comunicação segura e bi-direcional, mobilidade e tratamento de serviços de localização. Além disso, o padrão oferece suporte a IPv6, adaptação ao 6LoWPAN e funciona sobre a topologia estrela [5]. O fator atrativo do LoRAWAN é o seu baixo custo e a quantidade de empresas de hardware que estão o adotando (ALLIANCE, 2018). A taxa de comunicação alcança valores entre 300 bps a 50 kbps. O consumo de energia na LoRaWan é considerado pequena, o que permite aos dispositivos se manterem ativos por longos períodos. A LoRaWANs utiliza a frequência ISM sub-GHz fazendo com que as ondas eletromagnéticas penetrem grandes estruturas e superfícies, a distâncias de 2 km a 5 km em meio urbano e 45 km no meio rural. Os valores de frequência mais usadas pelo LoRaWan são: 109 MHz, 433 MHz, 866MHz e 915 MHz. O MTU adotado pelo padrão LoRaWAN é de 256 bytes [6].

Sigfox. O SigFox [7] utiliza a tecnologia Ultra Narrow Band (UNB), projetada para lidar com pequenas taxas de transferência de dados. Essa tecnologia ainda é bastante recente e já possui bastante aceitação, chegando a dezenas de milhares de dispositivos espalhados pela Europa e América do Norte. A SigFox atua como uma operadora para IoT, com suporte a uma série de dispositivos. A principal função é abstrair dificuldades de conexão e prover uma API para que os usuários implementem sistemas IoT com maior facilidade. O raio de cobertura oficialmente relatada, em zonas urbanas, está entre 3 km e 10 km e em zonas rurais entre 30 km a 50 km. A taxa de comunicação varia entre 10 bps e 1000 bps e o MTU utilizado é de 96 bytes. O SigFox possui baixo consumo de energia e opera na faixa de 900 MHz.

A Tabela 1 resume as tecnologias de comunicação apresentadas nesta seção. As principais características de cada uma são elencadas, o que permite compará-las. Em particular, destaca-se a grande variedade de possibilidades para conectar dispositivos. Portanto, é preciso ponderar acerca das características das tecnologias e finalidade do dispositivo para escolher a melhor forma de conectá-lo.

[5] https://www.semtech.com/products/wireless-rf/lora-transceivers
[6] https://lora-alliance.org
[7] https://www.sigfox.com/en/sigfox-developers

Tabela 1 – Comparação entre as tecnologias de comunicação.

Protocolo	**Alcance**	**Frequência**	**Taxa**	**IPv6**	**Topologia**
Ethernet	100/2000m	N/A	10 Gbps	Sim	Variada
Wi-Fi	50m	2.4/5 GHz	1300 Mbps	Sim	Estrela
BLE	80m	2.4 GHz	1 Mbps	Sim	Estrela/*Mesh*
ZigBee	100m	915 MHz/2.4 GHz	250 kbps	Sim	Estrela/*Mesh*
3G/4G	35/200 km	1900/2100/2500MHz	1/10 Mbps	Sim	Estrela
SigFox	10/50 km	868/902MHz	10–1000 bps	-	-
LoraWan	2/5 km	Sub-GHz	0.3-50 kbps	Sim	Estrela

Fonte: (PERES et al.,)

2.3.3 Protocolos de Comunicação

Na Internet tradicional existem diversos protocolos de comunicação responsáveis por gerenciar a transferência de dados em uma rede que conecta dois ou mais computadores. Para citar alguns exemplos: HTTP, FTP e SFTP. Quando se fala em comunicação entre dois ou mais dispositivos (ou um conjunto de aplicações) conectados em uma rede também surge a necessidade de pensar em um protocolo que gerencie esta comunicação, isto é, a troca de mensagens e/ou dados entre os elementos que compõe esta rede de objetos de forma eficiente considerando as características e limitações impostas pelo ambiente (MARTINS; ZEM, 2016). Neste cenário, surgem dois protocolos de transferência que podem ser utilizados em ambientes restritos (capacidade computacional limitada, rede com perdas, etc): o *Messaging Queue Telemetry Transport* (MQTT) e o *Constrained Application Protocol* (CoAP).

2.3.3.1 MQTT

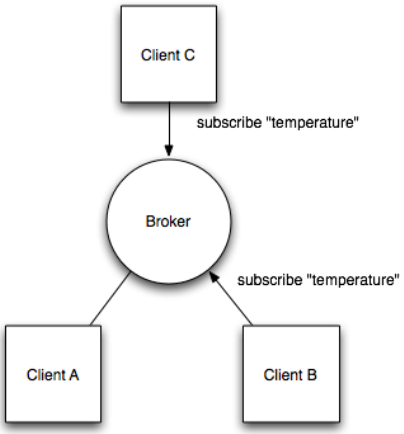
O MQTT foi criado em meados de 1999 por Andy Stanford-Clark (IBM) e Arlen Nipper (Eurotech). Trata-se de um protocolo de mensagens baseado na arquitetura *publish/subscribe*, voltado para dispositivos restritos e redes inseguras, com baixa largura de banda e alta latência. Segundo o projeto (MQTT.org, 1999), os princípios do design são minimizar os requerimentos de recursos de dispositivo e de largura de banda tentando garantir confiabilidade e garantia de entrega.

Atualmente na versão 3.1, a especificação do MQTT (EUROTECH; IBM, 2010) apresenta uma série de características do protocolo, algumas delas listadas abaixo:

- Uso de TCP/IP para fornecer conectividade;
- Pequena sobrecarga de transporte e trocas minimizadas de protocolos para reduzir tráfego na rede;
- Mecanismo que notifica partes interessadas quando um cliente se desconecta da rede anormalmente.

Segundo (JAFFEY, 2014), o protocolo segue o modelo cliente/servidor. Os dispositivos sensores são clientes que se conectam a um servidor (chamado de *broker*) usando TCP. As mensagens a serem transmitidas são publicadas para um endereço (chamado de tópico), que inclusive, assemelha-se a uma estrutura de diretórios em um sistema de arquivos, por exemplo, casa/quarto/temperatura. Clientes por sua vez podem se subscrever para vários tópicos, tornando-se assim capazes de receber as mensagens que outros clientes publicam neste tópico. A Figura 4 mostra uma rede conectando três clientes com um *broker* central.

Figura 4 – Subscrição de tópico ao broker

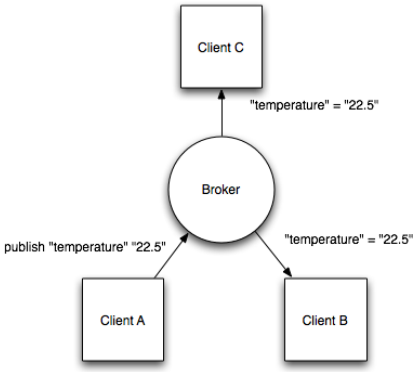


Fonte: (JAFFEY, 2014)

Conforme a Figura 4, todos os clientes conectam-se ao broker, e os clientes B e C subscrevem para o tópico temperatura. Posteriormente, o cliente A publica o valor '22.5' para o tópico temperatura. O *broker* então encaminha a mensagem para os clientes subscritos para aquele tópico, possibilitando assim, que os clientes comuniquem-se um-

a-um, um-para-muitos ou muitos-para-muitos. O processo de publicação da mensagem pode ser visto na Figura 5.

Figura 5 – Publicação de mensagem no broker.



Fonte: (JAFFEY, 2014)

Formato da Mensagem

Cada uma das chamadas mensagens de comando MQTT possui um cabeçalho fixo composto de dois *bytes*, onde o primeiro *byte* contém o campo que identifica o tipo da mensagem e também campos marcadores (DUP, nível QoS e RETAIN). A Tabela 2 mostra o formato do cabeçalho fixo das mensagens.

Tabela 2 – Cabeçalho fixo de uma mensagem MQTT.

bit	**7 6 5 4**	**3**	**2 1**	**0**
byte 1	Message Type	DUP flag	QoS level	RETAIN
byte 2	Remaining Length			

Fonte: (EUROTECH; IBM, 2010)

O tipo da mensagem ocupa do bit 7 ao 4 e ao primeiro *byte* do cabeçalho fixo e é representado por um valor não assinado. A lista de tipos definida na versão 3.1 do MQTT é descrita na Tabela 3. Os quatro bits restantes do primeiro *byte* dividem-se em quatro campos que servem de marcadores para indicar preferências definidas antes do envio da mensagem. São eles:

- *Duplicate delivery* (DUP): acrônimo relativo à entrega duplicada, este marcador ocupa o bit 4 e é ativado quando o cliente ou o ser-

vidor tentam reenviar mensagens do tipo PUBLISH, PUBREL, SUBSCRIBE ou UNSUBSCRIBE (ver tabela 1) que tenham *Quality of Service* (QoS) maior que 0 e requeiram *acknowlegment* (ACK);

- *Quality of Service* (QoS): este marcador ocupa os dois penúltimos bits e indica o nível de garantia da entrega de uma mensagem PUBLISH. Os níveis de QoS são mostrados na Tabela 3;

- RETAIN: quando um cliente envia uma mensagem PUBLISH ao servidor com este marcador ativado, ela deve ser retida no servidor mesmo depois de ser entregue aos assinantes. No evento de uma nova subscrição a um tópico, a última mensagem retida para este tópico deve ser enviada para o novo assinante caso este marcador esteja ativado.

Tabela 3 – Tipos de mensagem.

Mnemônico	**Enumeração**	**Descrição**
Reservado	0	Reservado.
CONNECT	1	Requisição de conexão do cliente ao servidor.
CONNACK	2	ACK de conexão.
PUBLISH	3	Publicação de mensagem.
PUBACK	4	ACK de publicação.
PUBREC	5	Publicação recebida (garantia de entrega parte I).
PUBREL	6	Publicação liberada (garantia de entrega parte II).
PUBCOMP	7	Publicação completa (garantia de entrega parte III).
SUBSCRIBE	8	Requisição de subscrição do cliente.
SUBACK	9	ACK de subscrição.
UNSUBSCRIBE	10	Requisição de cancelamento de subscrição do cliente.
UNSUBACK	11	ACK de cancelamento de subscrição.
PINGREQ	12	Requisição PING.
PINGRESP	13	Resposta PING.
DISCONNECT	14	Cliente desconectando.
Reservado	15	Reservado.

Fonte: (EUROTECH; IBM, 2010)

A largura restante do cabeçalho fixo (*byte* 2) é usada para representar nada mais do que a quantidade de *bytes* remanescentes na mensagem. Incluindo dados do cabeçalho variável e do *payload*. Cabeçalho variável é um componente presente em alguns tipos de mensagem MQTT e está localizado entre o cabeçalho fixo e o *payload*. Usado principalmente nas mensagens CONNECT, este cabeçalho possui dois campos para nome e versão do protocolo, respectivamente e mais uma série de marcadores que definirão algumas diretivas para a conexão entre cliente e servidor.

Tabela 4 – Níveis de QoS.

Valor QoS	Bit 2	Bit 1	Descrição		
0	0	0	Até uma vez	Disparar e esquecer	<1
1	0	1	Ao menos uma vez	Entrega com ACK	>1
2	1	0		Entrega garantida	1
3	1	1	Reservado		

Fonte: (EUROTECH; IBM, 2010)

2.3.3.2 CoAP

O CoAP foi criado por um grupo de trabalho do *Internet Engineering Task Force* (IETF) chamado *Constrained RESTful Environments* (CoRE). O grupo iniciou suas atividades em março de 2010 com objetivo de prover um *framework* para aplicações que manipulam recursos simples localizados em dispositivos interligados em redes limitadas, incluindo desde aplicações que monitoram sensores de temperatura e medidores de energia, até controle de atuadores como interruptores ou trancas eletrônicas, e também aplicações que gerenciam os dispositivos que compõem a rede, conforme o (IETF, 2010). O CoAP trata de uma parte deste *framework*. Sendo assim, a definição do protocolo foi um dos primeiros marcos definidos pelo CoRE. O primeiro *draft* para o CoAP foi publicado no IETF em junho de 2010.

De acordo com o RFC 7252 (IETF, 2014, p. 1), o CoAP é um protocolo de transferência voltado para nós e para redes restritas (considerando nós com pequena quantidade de memória RAM e redes em que a taxa de perda de pacotes é alta). O protocolo foi projetado para aplicações M2M como, por exemplo, *smart energy* e automação residencial. Ele define quatro tipos de mensagens: *Confirmable*, *Non-confirmable*, *Acknowledgment* e *Reset* (IETF, 2014, p. 8).

- ***Confirmable*** (CON), como o nome sugere, são mensagens que precisam ser confirmadas no destino. Assim, quando não há perda de pacotes, cada mensagem deste tipo resulta em uma mensagem do tipo *Acknowledgment* ou *Reset*;

- ***Non-confirmable*** (NON), não necessitam de confirmação de recebimento. Esta característica é útil no caso de uma aplicação que recebe leituras constantes de um sensor de temperatura em um espaço muito curto de tempo, onde a perda de uma ou outra mensagem não é motivo para preocupações;

- ***Acknowledgment*** são mensagens que confirmam o recebimento de uma mensagem *Confirmable*. É importante ressaltar que por si só, uma mensagem ACK não indica sucesso ou falha de nenhuma requisição encapsulada na mensagem *Confirmable*;

- ***Reset*** indicam que outra mensagem (CON ou NON) foi recebida, mas por falta de algum contexto ela não pôde ser devidamente processada. Ela pode ocorrer no caso de algum dispositivo ter reiniciado e a mensagem enviada não foi devidamente interrompida.

O CoAP provê um modelo de interação requisição/resposta entre os endpoints das aplicações, suporta descoberta embutida de serviços e recursos e inclui conceitos chave da Web como por exemplo *Uniform Resources Identifiers* (URIs) e tipos de mídia comuns na Internet. Além disso, o protocolo foi projetado para interagir com o HTTP para integração com a *Web*. Abaixo estão algumas características descritas no RFC 7252:

- Protocolo *Web* que cumpre com os requerimentos M2M em ambientes restritos;

- Troca de mensagens assíncrona;

- Suporte à URI e *Content-Type*;

- Capacidades simples de *proxy* e *caching*;

- Mapeamento HTTP que permite que proxies possam prover acesso aos recursos do CoAP via HTTP de maneira uniforme;

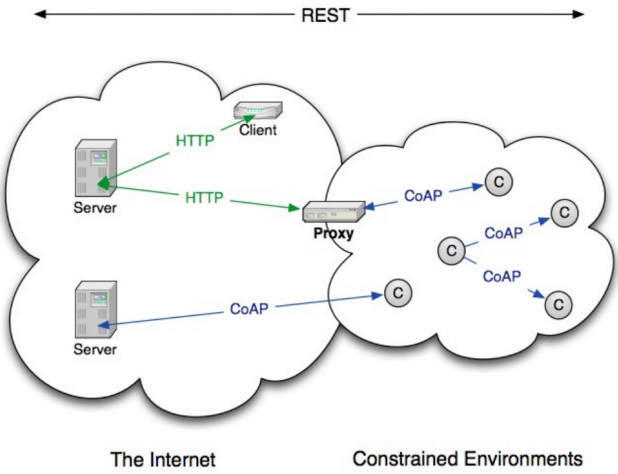
- Interligação segura para *Datagram Transport Layer Security* (DTLS);

- *Binding* em *User Datagram Protocol* (UDP) com confiabilidade opcional suportando requisições tanto *unicast* quanto *multicast*;

- Suporte aos métodos *GET, POST, PUT, DELETE.*

A Figura 6 ilustra a arquitetura CoAP em uma perspectiva de alto nível. Um dos objetivos do CoRE também consiste em adequar a arquitetura *Representational State Transfer* (REST), proposta por (FIELDING, 2000) para ambientes restritos (nós e rede). O CoAP foi desenvolvido de acordo com esta arquitetura, logo pode ser considerado como um protocolo RESTful, de acordo com (SHELBY, 2014).

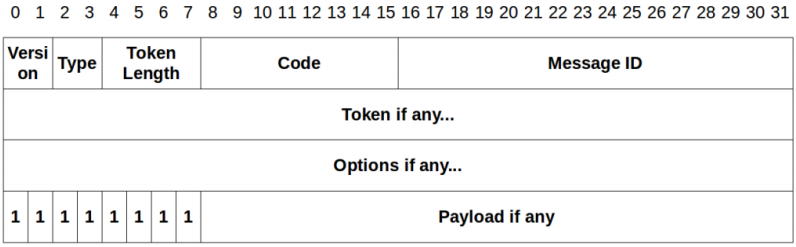
Figura 6 – Arquitetura CoAP.



Fonte: (SHELBY, 2014)

Formato da mensagem CoAP é baseado na troca de mensagens compactas que são transportadas sobre UDP. Suas mensagens estão codificadas em um formato binário com um cabeçalho de 4 *bytes*, seguido por um *token* de largura variável (de 0 a 8 bytes) . Após o *token* pode não haver nenhuma ou uma série de opções do CoAP no formato *Type-Length-Value* (TLV), que são opcionalmente seguidas por um *payload*, ocupando o resto do datagrama. A Figura 7 mostra o formato da mensagem CoAP.

Figura 7 – Formato da mensagem CoAP.



Fonte: (SHELBY, 2014)

Os campos que formam o cabeçalho do datagrama são definidos pelo RFC da seguinte maneira:

- Versão (ver): inteiro não assinado de dois bits que indica o número correspondente à versão do CoAP;
- Tipo (T): inteiro não assinado de dois bits que indica se o tipo da mensagem, que pode variar entre Confirmable (0), Non-confirmable (1), Acknowlegment (2) ou Reset (3);
- Largura do token (TKL): inteiro não assinado de quatro bits usado para indicar a largura variável do token (até 8 bytes);
- Código: inteiro não assinado de oito bits que caracteriza o tipo da mensagem, podendo indicar um método de requisição (Tabela 5) ou um código de resposta (Tabela 6);
- ID da mensagem: inteiro não assinado de 16 bits usado para detectar duplicação de mensagens e também para comparação de mensagens do tipo ACK, por exemplo.

Tabela 5 – Códigos dos métodos CoAP.

Código	Nome
0.01	GET
0.02	POST
0.03	PUT
0.04	DELETE

Fonte: (SHELBY, 2014)

Tabela 6 – Códigos de resposta CoAP.

Código	Nome
2.01	Criado
2.02	Deletado
2.03	Válido
2.04	Alterado
2.05	Conteúdo
4.00	*Bad Request*
4.01	Não autorizado
4.02	*Bad Option*
4.03	Proibido
4.04	Não encontrado
4.05	Método não permitido
4.06	Inaceitável
4.12	Precondição falhou
4.13	Entidade de requisição muito grande
4.15	*Context-format* não suportado
5.00	Erro interno no servido
5.01	Não implementado
5.02	*Bad gateway*
5.03	Serviço indisponível
5.04	*Gateway timeout*
5.05	*Proxying* não suportado

Fonte: (SHELBY, 2014)

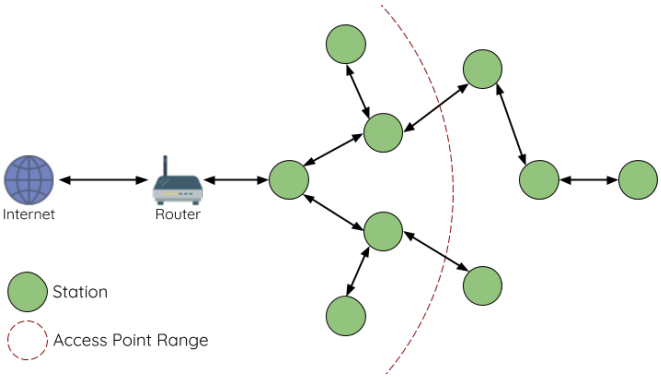
O *token* é utilizado para relacionar requisições e respostas. Ele é gerado no cliente e transportado junto com a requisição. O servidor deve então ecoar este *token* na resposta correspondente.

Conforme citado anteriormente, seguindo o cabeçalho, o *token* e as opções (uso opcional), vem o *payload*, que é um campo opcional. Quando a mensagem possui *payload* ele deve ser indicado por um marcador de um *byte* (*Payload Marker*) que indica o fim das opções e o início da carga. A ausência do marcador significa um *payload* de largura 0 *bytes*, enquanto a presença de um marcador seguido por um *payload* de 0 *bytes* deve ser processado como um erro no formato da mensagem.

2.3.4 Redes *Mesh*

Redes Wireless *mesh*, ou *Wireless mesh Networks* (WMN), são casos específicos de redes ad hoc. Segundo (RAMANATHAN; REDI, 2002), uma rede ad hoc (possivelmente móvel) é um conjunto de dispositivos de rede que pretendem se comunicar, mas que não possuem infra-estrutura fixa disponível e não possuem organização pré-determinada de links de comunicação disponíveis. Os nós individuais da rede são responsáveis por descoberta dinâmica de quais são os outros nós que podem se comunicar diretamente a ele, ou seja, de quais são seus vizinhos (formando uma rede multi-hop) (JOSÉ; JR; PRZYBYSZAUTOR, 2007) como mostrado na Figura 8. Redes ad hoc são escolhidas para serem usadas em situações onde a infra-estrutura não está disponível ou não é confiável, ou ainda em situações de emergência. No caso deste trabalho, o problema vem a ser em relação ao monitoramento do consumo de energia elétrica, cuja avaliação das características elétricas da rede onde, dependendo do ambiente, é importante o monitoramento contínuo dos dados.

Figura 8 – Arquitetura de Rede tipo *Mesh*.



Fonte: (ESPRESSIF, 2017)

A rede *mesh* traz algumas vantagens em relação às redes tradicionais que possuem um nó central (TEIXEIRA, 2004):

- Aumento da distância entre a origem e o destino, sem prejudicar a taxa de transmissão pois é possível sair de qualquer nó e chegar em outro sendo roteado pelos demais;
- Otimização do espectro de frequências pois tendo os nós uma maior proximidade, a potência do sinal é menor;
- Não necessidade de linha de visada já que em uma rede *mesh* todos os nós se comunicam e também evitam obstáculos;
- Redução do custo da rede com a utilização de equipamentos considerados terminais como se fossem roteadores e/ou repetidores;
- Redução da necessidade de conexões entre os Access Points e a Internet devido à utilização de equipamentos que somente roteiam ou repetem a informação sendo necessário assim menos equipamentos que tenham acesso direto à internet;
- Robustez.

No caso das redes inteligentes como redes de *Smart Meters* e *Smart Grids*, é possível utilizar uma rede *mesh* para interligar os medidores de um estabelecimento ou pontos estratégicos de medição de uma rede, ou seja, os nós formando uma rede de comunicação onde os dados de todos os nós podem ser transmitidos até a borda da rede e

ligado diretamente com a concessionária ou servidor em local/nuvem para acesso as informações.

2.3.5 Protocolos de Roteamento para Redes *Mesh*

Algoritmo de roteamento é "a parte do *software* da camada de rede responsável pela decisão sobre a linha de saída a ser usada na transmissão do pacote de entrada" (TANENBAUM, 1996). Para esta tomada de decisão são levados em consideração alguns fatores como, por exemplo, o número de saltos e o estado do link.

Existem três tipos de protocolos de roteamento: pró-ativos, reativos e híbridos. O pró-ativo exige que os nós da rede mantenham a rota de todos possíveis destinos de modo que, quando houver necessidade do envio de um pacote de dados, a rota seja conhecida para ser usada, imediatamente. Já nos protocolos reativos, os nós descobrem os destinos sob-demanda, ou seja, não necessitam de uma rota para os destinos até que precisem enviar pacotes de dados para os destinos (FARIAS et al., 2005). Os protocolos híbridos são aqueles que apenas um conjunto de nodos realiza atualização periódica sobre as informações de possíveis nodos destinos, tentando fazer uso conveniente das duas abordagens anteriores (YANG; TSENG, 2004).

Para cada tipo de aplicação um protocolo de roteamento específico deve ser escolhido para que atenda as características da rede, portando, deve-se levar em conta alguns fatores na hora da escolha, como: a distância em saltos da origem até o destino, o custo de processamento envolvido, o numero de nós da rede, o tempo de atualização da tabela de roteamento (levando em consideração o numero de mensagens trocadas para este caso) e o consumo energético.

2.3.5.1 OLSR

De acordo com (SILVA, 2015), o protocolo OLSR (Optimized Link State Routing Protocol) foi criado para ser utilizado em grandes redes Ad Hoc,calculando e mantendo rotas para todos os dispositivos de uma rede construída sob uma topologia em malha.

Por ser um protocolo pró-ativo, o OLSR tem como característica principal, a atualização constante das tabelas de roteamento de todos os nós participantes da rede. Dessa forma, qualquer modificação na topologia da rede é detectada automaticamente. Nesse contexto, o pro-

tocolo OLSR utiliza mensagens de controle para realizar a descoberta de nós na rede e atualização da topologia (SILVA, 2015). Essas mensagens são enviadas em broadcast através da porta UDP 698 (CLAUSEN; JACQUET, 2003).

De acordo com (CLAUSEN; JACQUET, 2003) e (FERNANDES et al., 2006), as seguintes mensagens são essenciais para o funcionamento do protocolo OLSR:

- HELLO: utilizada para realizar a escolha de um MPR e a descoberta de enlaces e vizinhos. Essa mensagem é enviada apenas para os vizinhos e não deve ser encaminhada para os vizinhos distantes a mais de um salto e apresenta os nós com os quais um nó possui uma comunicação.

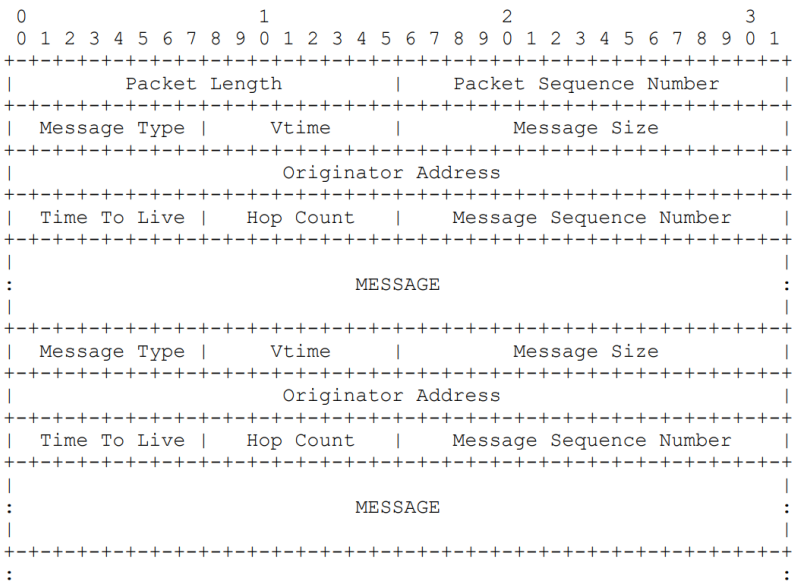
- TC: é utilizada para realizar o controle da topologia da rede. Essa mensagem é composta por uma lista de nós que selecionaram o nó emissor dessa mensagem como MPR e enviada para toda a rede. Dessa forma, é possível que um nó não envie essa mensagem, em razão não ter sido escolhido como MPR. As informações recebidas através dessa mensagem são armazenadas na tabela de roteamento de cada nó, a qual é utilizada para realizar o cálculo de rota para enviar um pacote a um determinado destino.

- MID: essa mensagem é enviada para toda a rede e utilizada para declarar múltiplas interfaces em um nó. Através dessa mensagem, é possível associar diversas interfaces de um dispositivo no cálculo de rotas.

- HNA: contém informações sobre os anúncios das redes de um nó, ou seja, um nó apresenta-se como gateway para o acesso de uma determinada rede.

Conforme (CLAUSEN; JACQUET, 2003), o protocolo OLSR utiliza um formato de pacote unificado para todas as informações relacionadas ao protocolo. Dessa forma, cada pacote pode encapsular mais de uma mensagem. Essas mensagens compartilham um formato de cabeçalho comum. A Figura 9 apresenta o formato de um pacote OLSR. Nela podemos visualizar o cabeçalho do pacote e o encapsulamento de mensagens.

Figura 9 – Formato dos Pacotes OLSR.



Fonte: (CLAUSEN; JACQUET, 2003)

2.3.5.2 AODV

Trata-se de um protocolo reativo (on-demand) puro, pois não é baseado na manutenção de informações sobre rotas pelos nós, tampouco na troca de informações de roteamento entre os nós (PERKINS; BELDING-ROYER; DAS, 2003).

Apesar de sua natureza sob-demanda, o AODV precisa do conhecimento da conectividade local para operar. Para tanto, um mecanismo periódico de broadcast local através das mensagens de hello pode ser utilizado, ou o AODV pode apoiar-se em mensagens de camada MAC para identificar os nós vizinhos. O AODV utiliza, ainda, números de sequencia para evitar a formação de loops (FIGUEIREDO et al., 2006).

Quando é necessária a descoberta de uma rota, o nó fonte envia em broadcast um pacote route request (RREQ) contendo, entre outros campos, o endereço do nó fonte e um identificador broadcast id que juntos identificam unicamente um procedimento de descoberta de rota. Sempre que uma fonte inicia um novo procedimento de descoberta de rota, o valor de broadcast id é incrementado. Os nós intermediários vão

incrementando o contador de saltos do RREQ e reenviando o RREQ para seus vizinhos, ate que este atinja o destino ou algum nó intermediário que possua uma entrada valida em sua tabela de roteamento para o nó destino. Ao reenviar um RREQ para seus vizinhos o nó intermediário precisa armazenar o endereço IP da fonte e do destino, o broadcast id, o tempo de vida do caminho reverso e o numero de sequencia da fonte, para o caso de uma eventual requisição de rota para o nó fonte, que pode ser satisfeita por esta rota reversa. Cada nó intermediário processa apenas uma vez cada RREQ, descartando RREQs redundantes.

Para um nó intermediário estar apto a responder um RREQ, ele precisa ter uma entrada válida na tabela de roteamento com numero de sequencia do destino mais recente que o enviado pela fonte. Para satisfazer um RREQ, o nó intermediário com a entrada ativa na tabela ou o nó destino envia em unicast um route reply (RREP) com, entre outros campos, o contador de saltos e o número de sequencia conhecido para o destino. 'A medida que o RREP vai retornando pelo caminho reverso do RREQ original, ele vai estabelecendo apontadores para o nó que o enviou. Ao receber um RREP redundante, o nó apenas o propaga se este contiver um numero de sequencia do destino maior que os anteriores ou o mesmo numero de sequencia com um contador de saltos menor.

A tabela de roteamento possui no máximo uma entrada para cada destino, e cada entrada esta associada a um route cache timeout que é o tempo apos o qual a entrada será considerada invalida. Cada entrada possui, ainda, uma lista com todos os vizinhos ativos através dos quais são recebidos pacotes para o destino em questão. O numero de sequencia das entradas ativas da tabela previnem a formação de loops.

É importante ressaltar que o protocolo AODV foi projetado para reduzir a disseminação de tráfego de controle, eliminando parcialmente o overhead no tráfego de dados, com o objetivo de aumentar a escalabilidade e o desempenho. Em contrapartida, por ser um protocolo reativo, existe a ocorrência de latências na transmissão do tráfego de dados para rotas novas na rede (FIGUEIREDO et al., 2006).

2.3.6 Formato JSON e Armazenamento

JSON (Javascript Object Notation) é um formato de serialização de dados legível por humanos baseado em texto com especificação

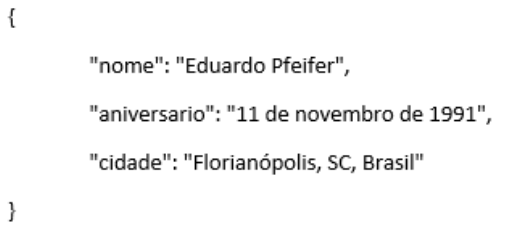
padronizada e parcialmente descritivo. Foi desenvolvido por Douglas Crockford com o objetivo de representar dados de uma maneira simples, leve e flexível através da redução na sobrecarga de marcações comparado ao formato XML. Por ter se adaptado bem no ambiente de aplicações distribuídas, este formato acabou sendo amplamente utilizado em serviços como principal forma de representação de dados serializados (DUVANDER, 2013).

Por ter se adaptado bem no ambiente de aplicações distribuídas, este formato acabou sendo amplamente utilizado em serviços como principal forma de representação de dados serializados (DUVANDER, 2013).

Neste trabalho por exemplo, foi utilizado para o envio dos dados entre os nodos da rede e também para o servidor responsável pelo armazenamento dos dados.

Através da composição de listas, objetos e tipos primitivos, consegue-se representar complexas estruturas de dados. Não existe, no entanto, um único padrão de representação. Dada uma estrutura, é possível representá-la de inúmeras maneiras. A seguir, está uma das formas distintas de representação em JSON de uma entidade “pessoa”:

Figura 10 – Representação de uma entidade "pessoa"em JSON.



Fonte: Do Autor

Para que a grande quantidade de dados gerados pelos sensores da IoT possa ser posteriormente analisada e processada, a etapa de armazenamento faz-se essencial (PERES et al.,). Como apresentado na Figura 7, uma das principais formas de acesso a estes dados, e a mais comumente encontrada na prática, acontece por meio de servidores de armazenamento. Muitos destes estão disponíveis sob a forma de plataformas computacionais especificamente voltadas para prover serviços para a IoT. Na Tabela 7 apresentamos algumas plataformas para IoT

atualmente disponíveis, destacando algumas das suas principais características.

Além do armazenamento, algumas plataformas também disponibilizam outros serviços, como a marcação de tempo (*timestamp*) de todos os dados recebidos, algumas funcionalidades de processamento, que geralmente são pré-definidos e acessados sob a forma de *wizards* ou *dashboards*, definição de regras para a execução de atividades com base em eventos ou comportamento dos dados, entre outros. Para mais detalhes sobre o projeto e implementação de plataformas para IoT, ver o trabalho de (DELICATO, 2015).

Tabela 7 – Algumas das principais plataformas para IoT.

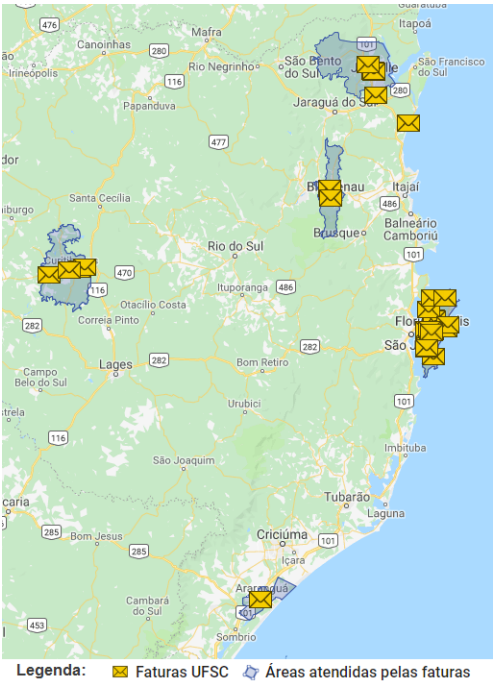
Plataforma	Endereço	Descrição	Tipo de conta
AWS IoT	aws.amazon.com/iot	Plataforma da Amazon voltada para empresas.	Possui conta gratuita, mas pede cartão de crédito para confirmar.
Carriots	carriots.com	Plataforma com foco empresarial. Provê serviços de gerenciamento de e comunicação entre dispositivos	Possui conta gratuita, mas com várias limitações, e.g., número de dispositivos e acessos.
IBM Bluemix	bluemix.net	Plataforma da IBM com foco empresarial.	Possui conta gratuita, mas pede cartão de crédito para confirmar.
Microsoft Azure IoT	microsoft.com/iot	Plataforma da Microsoft voltada a IoT com foco empresarial.	Possui conta gratuita de um mês para testes.
OpenSensors	opensensors.io	Plataforma robusta com o foco principal para sensores abertos. Permite publish-subscribe para disponibilização dos dados de sensores	Conta gratuita disponível, paga-se apenas para ter sensores privados.
ThingSpeak	thingspeak.com	Plataforma robusta, com várias funcionalidades, como sensores públicos e busca por histórico	Conta gratuita disponível.

Fonte: (PERES et al.,)

3 SITUAÇÃO ATUAL DO CONSUMO ENERGÉTICO NA UFSC

O Departamento de Projetos de Arquitetura e Engenharia (DPAE) da UFSC elaborou um mapa que apresenta a média do consumo global de energia elétrica em regiões de todos os campi da Universidade onde seus indicadores sejam atualizados mensalmente como mostrado na Figura 11. Os estudos foram baseados nas faturas de energia elétrica da UFSC a partir de 2013.

Figura 11 – Localização das faturas e áreas atendidas pela UFSC.



Fonte: Adaptado (UFSC, 2014)

O DPAE trabalha no monitoramento e mapeamento das redes de energia elétrica de média tensão – redes subterrâneas de energia dentro dos campi – e da rede de água. “O objetivo é monitorar para que se possa, de um lado, buscar e avaliar as ações de eficiência aplicadas e, de outro, acompanhar mensalmente os contratos junto aos fornecedores

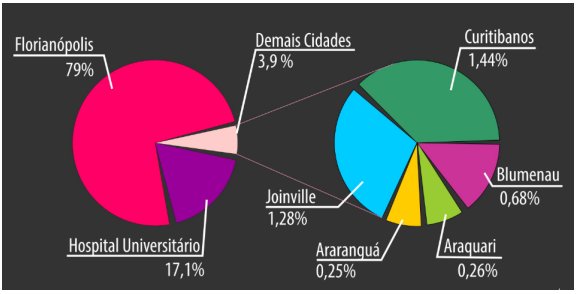
– alterando-os de acordo com o que preveem as normativas da Agência Nacional de Energia Elétrica sempre que sejam vantajosos para a UFSC, principalmente, nos requisitos de escolha da demanda e da modalidade tarifária a ser contratada", explica o engenheiro Irvando Luiz Speranzini (BERTOLDI, 2014).

Conforme o mapeamento, o maior consumo está no campus do bairro Trindade. O Hospital Universitário (HU), também em Florianópolis, está em segundo lugar. "O consumo na UFSC está dentro da média, mas tende a em aumentar em função da quantidade de obras que vêm sendo desenvolvidas na Universidade", conclui Irvando Luiz Speranzini.

3.1 NÚMEROS SOBRE O CONSUMO

Como mostradona Figura 12, do total consumido nas 82 faturas pagas à CELESC (Centrais Elétricas de Santa Catarina S.A.), 96% concentram-se em Florianópolis – com a expansão dos Campi, espera-se que este quadro se modifique futuramente.

Figura 12 – Distribuição de consumo de energia elétrica pela UFSC.



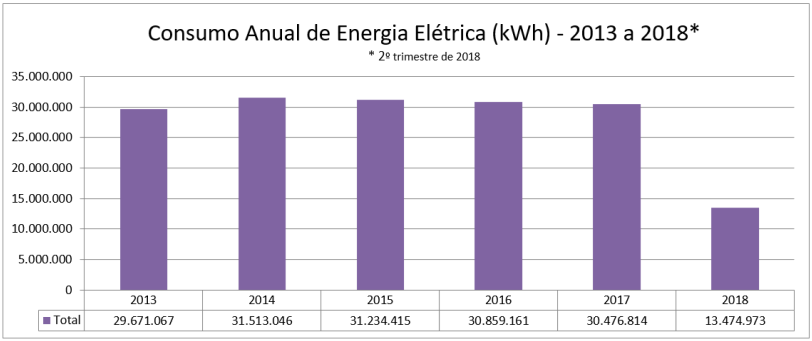
Fonte: Adaptado (UFSC, 2017)

O número de unidades consumidoras em alta tensão, que exigem uma contratação diferenciada, também aumentou: em 2015 eram 19; em 2016, terminou-se com 23 unidades com atendimento em tensão de 2,3 kV a 44 kV (UFSC, 2017).

Como mostrado na Figura 13, entre os anos de 2013 a 2018 o maior consumo de energia elétrica em quilowatt-hora (kWh) na UFSC ocorreu em 2014 com 31.513.046 kWh consumidos, em seguida, 2015 com uso de 31.205.492 kWh e 2016 com 30.848.271 kWh consumidos,

valor capaz de abastecer 14,2 mil residências catarinense por um ano (UFSC, 2017).

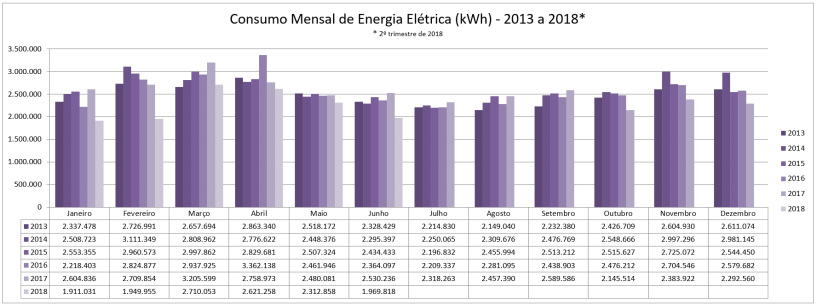
Figura 13 – Consumo anual de energia.



Fonte: (UFSC, 2013a)

Já em relação a Figura 14 que apresenta o consumo mensal de 2013 à 2018, os meses novembro, dezembro e fevereiro à abril apresentaram maior consumo de energia em kWh, enquanto o período de menor consumo foi de maio a outubro e janeiro.

Figura 14 – Consumo mensal de energia.



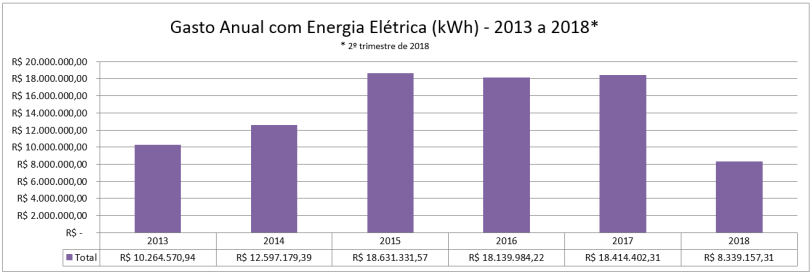
Fonte: (UFSC, 2013a)

Em relação ao ano de 2018, observa-se que o primeiro trimestre do ano teve consumos mensais inferiores a média de cada mês, com exceção do mês de Março (2.710.053 kWh), que pode ser justificado pelo fato de ser o primeiro mês completo de aulas na Universidade e consequente aumento de pessoas no campus.

3.2 GASTOS FINANCEIROS

Quanto aos gastos financeiros com energia elétrica nesse mesmo período (2013 - 2018), observou-se que 2015 foi o ano com maior valor despendido, R$ 18.631.261,06, mesmo apresentando menor consumo de energia em kWh em relação a 2014 como mostrado na Figura 15. Isso se deve ao fato de que, em 2015, houve um aumento de 50% do valor em reais de kWh, resultando nesta discrepância de valores. Em 2017 houve o segundo maior valor despendido em energia elétrica totalizando R$ 18.414.402,31. O ano de 2016 gerou uma despesa próxima à 2015 de R$ 18.139.984,22, enquanto em 2013 e 2014 o valor gasto foi de R$ 10.264.570,94 e R$ 12.597.179,39, respectivamente (UFSC, 2013a).

Figura 15 – Gasto anual de energia



Fonte: (UFSC, 2013a)

4 MONITORAMENTO DO CONSUMO

4.1 MEDIDORES INTELIGENTES

A implementação de redes e medidores inteligentes (*Smart Meters*) pode ser uma possível solução para a redução da demanda de energia, gestão eficiente na geração e otimização da gestão de recursos. Os medidores inteligentes tem capacidade de realizar uma sofisticada medição, cálculos, calibração e comunicação através de *hardware* e *software*. Para interoperabilidade dentro da infraestrutura dessa rede, medidores inteligentes são projetados para realizar funções programáveis, armazenar e transmitir dados de acordo com determinados padrões (BARAI; KRISHNAN; VENKATESH, 2015). Podemos citar as três principais vantagens dos *smart meters*:

Os medidores inteligentes podem fornecer informações, em maior frequência, sobre período pico ou fora de pico e padrão de uso de energia utilizando comunicação bidimensional entre o sistema de ponta e gestor do consumidor. Os consumidores podem reduzir as contas de energia elétrica e os distribuidores podem gerir melhor o seu mercado. Portanto, a utilização do medidor inteligente está aumentando em um ritmo elevado (ZHENG; GAO; LIN, 2013)

- Medidores inteligentes economizam tempo: As leituras de um medidor são enviadas eletronicamente, significando que não há nenhum gasto de logística quando se precisa de uma leitura. Não é preciso que ninguém estranho ou as vezes com falta de preparo vá até as instalações, geralmente sujas, para encontrar o medidor. Ou você não terá que deixar um estranho entrar para ler seu medidor.

- Medidores inteligentes dão mais controle sobre o modo como a energia é usada: Usando medidores inteligentes, é possível acompanhar o gasto de energia em reais e centavos, kilowatt horas, ou mesmo medir emissões de carbono. É possível descobrir o que acontece com o uso de energia em horários de pico e se programar melhor para uma economia de energia inteligente. Uma vez que é possível ver como está sendo usada energia, pode-se então começar a controlar melhor o orçamento.

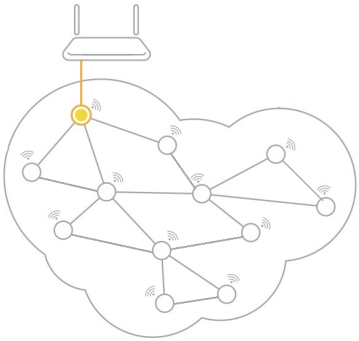
- Os contadores inteligentes ajudam a reduzir as contas de energia: Medidores inteligentes não reduzem diretamente a conta de ener-

gia. Mas eles podem lhe dar muitas informações inteligentes para ajudar a reduzir custos. Pode-se definir orçamentos diários para que saiba o quanto está sendo gasto e quando atingi-se limites diários. Além de poder descobrir quando uma casa está começando a usar muita energia, então pode-se mudar algumas coisas para economizar dinheiro.

4.1.1 Medidores Inteligentes e Redes *mesh*

O medidor inteligente coleta dados medidos no consumidor final e os transmite através de uma rede de rádio sem fio até o concentrador de dados que os envia ao distribuidor. O dado então é processado por meio de vários métodos com a finalidade de faturamento, administração do sistema, recuperação em caso de falhas e outros objetivos tanto técnicos quando operacionais (ZHENG; GAO; LIN, 2013). O medidor se comunica com outro por meio de uma LAN, caso existam vários medidores localizados numa mesma área eles podem trocar informações entre si. A Figura 16 mostra que todos os medidores estão conectados e trocam informações. Caso um medidor não consiga se comunicar com outro, devido a distância entre eles, o dado é passado por outros medidores até que eles possam se comunicar (ZHENG; GAO; LIN, 2013).

Figura 16 – Exemplo de Rede *Mesh*.



Fonte: Adaptado (ESPRESSIF, 2018)

5 AMBIENTES DE DESENVOLVIMENTO

5.1 MICROCONTROLADORES

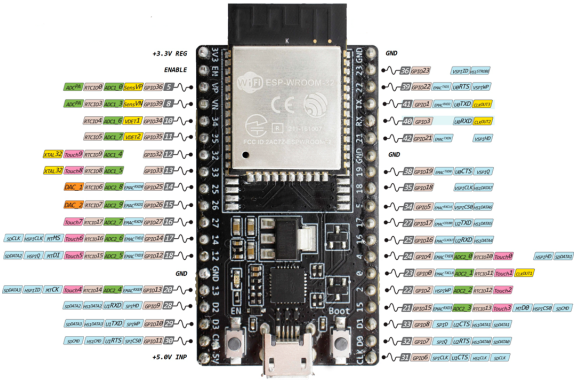
O microcontrolador é um dispositivo semicondutor em forma de circuito integrado (CI) que possui memórias não voláteis e voláteis, e portas de entrada e de saída digitais e analógicas. Normalmente, é utilizado em tarefas específicas – como na automação de sistemas, que não exijam grandes quantidades de dados pois tem pouca memória de armazenamento.

5.1.1 ESP32

O ESP32 é um microcontrolador desenvolvido pela empresa Espressif e constitui um microprocessador dual core Tensilica Xtensa de 32 bits com suporte embutido à rede Wi-Fi (802.11) e bluetooth versão 4.2, e com a memória flash integrada. Nesta plataforma, desenvolvida a partir de 2017,a programação pode ter mais de uma interface, como IDE (Arduino).

A plataforma NodeMCU-32S é uma placa baseada no ESP32 e é voltada para prototipagem IoT. Esse projeto, *open source* ou código aberto, consiste em um modelo de desenvolvimento que promove um licenciamento livre da esquematização de um produto, e a redistribuição universal desse esquema oferece a possibilidade para que qualquer um consulte, examine ou modifique o produto, e também possa aprimorá-lo. O microcontrolador possui 36 GPIOs (entradas ou saída) e 18 entradas A/D e duas saídas D/A, conforme mostra a Figura 17.

Figura 17 – Diagrama NodeMCU-32S.



Fonte: Adaptado (MURTA, 2018)

As vantagens e características técnicas do NodeMCU-32S são:

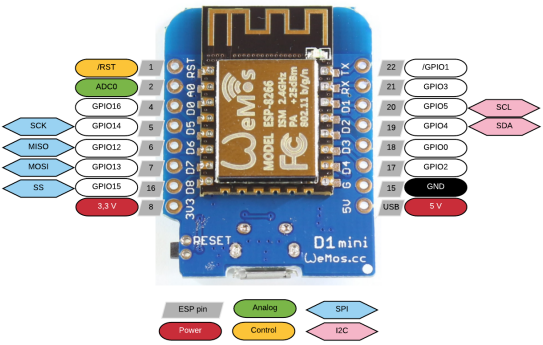
- Baixo custo;
- Suporte integrado para rede Wi-Fi 802.11 b/g/n e bluetooth versão 4.2;
- Tamanho reduzido da placa;
- Baixo consumo de energia, alto desempenho;
- Amplificador de baixo ruído, robustez, versatilidade e confiabilidade. Entre as desvantagens do ESP-WROOM-32, podem-se citar:
- Necessidadede aprender uma nova linguagem por ser um projeto recente;
- Pinagem reduzida;
- Documentação escassa, pois essa placa foi lançada em 2017.
- É baseado no SoC (System on Chip) ESP32-D0WDQ6, módulo controlador ESP-WROOM-3;
- Tem microprocessador dual core Tensilica Xtensa 32-bit LX6, Clock ajustável de 80 MHz até 240 MHz;
- É composto pela memória ROM de 448 KB, tem SRAM de 520 KB, RTC Slow SRAM de 8 KB, RTC, SRAM de 8 Kb;

- Dispõe de interfaces de GPIO, Sensores capacitivos, A/D, D/A, LNA pré- amplificado, CAN;
- Apresenta 36 GPIOs, GPIOs com função PWM / I2C e SPI.Possui A/D (conversor analógico digital) de 18 canais com resolução de 12 bits;
- Suporte a redes Wi-Fi padrão 802.11 b/g/n, possuindo opções de segurança WPA / WPA2 / WPA2 - Enterprise / WPS;
- Possui bluetooth 4.2 BLE;
- Possui 2 D/A (conversor digital analógico) com resolução de 8 bits.

5.1.2 ESP8266

Com o intuito de desempenhar o papel de ponte (Bridge) foi escolhido o microcontrolador *ESP*8266 também desenvolvido pela empresa Espressif, cujo modelo de placa utilizado é o WeMos D1 Mini representado na Figura 18. O ESP8266 possui um valor abaixo do ESP32 e cumpre os requisitos mínimos para que sua função seja realizada.

Figura 18 – Diagrama módulo WeMos D1 Mini.



Fonte: (BEST, 2018)

Especificações Técnicas:

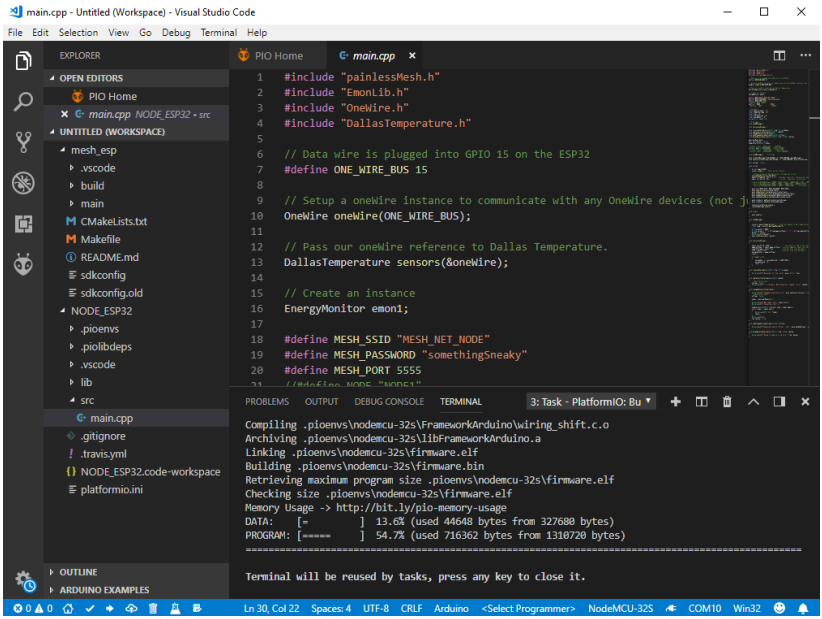
- Trabalha nas temperaturas: $-40\,^{\circ}\mathrm{C}$ até $+125\,^{\circ}\mathrm{C}$

- Peso 7 g;
- 32 KBytes de RAM para instruções
- 96 KBytes de RAM para dados
- 64 KBytes de ROM para boot
- É um System-On-Chip com $Wi\text{-}Fi$ embutido
- CPU que opera em 80 MHz, com possibilidade de operar em 160MHz
- Arquitetura RISC de 32 bits
- Tensão de operação: 3,3 V
- Suporte à redes: 802.11 b/g/n
- Alcance: 90 m aprox
- Suporta comunicação TCP e UDP
- Conectores: GPIO, I2C, SPI, UART, Entrada ADC, Saída PWM e Sensor de Temperatura interno.
- Modo de segurança: $OPEN/WEP/WPA_{PSK}/WPA2_{PSK}$

5.2 IDE

O Ambiente de Desenvolvimento Integrado (IDE, do inglês Integrated Development Environment) utilizado é o PlatformIO IDE para VSCode. O PlatformIO IDE é um ecossistema de código-aberto para desenvolvimento em IoT. Tem suporte para diversos tipos de placas, entre elas Arduino, Raspberry Pi e ESP32 (utilizado neste trabalho). Permite, portanto, que se use uma única ferramenta de desenvolvimento para diferentes microcontroladores. É escrito em Python e não necessita de nenhuma biblioteca ou ferramenta adicional.

Figura 19 – Visual Studio Code.



Fonte: Do Autor

O *Visual Studio Code* é um editor de código-fonte desenvolvido pela *Microsoft* para Windows, Linux e macOS. Ele inclui suporte para depuração, controle Git incorporado, realce de sintaxe, complementação inteligente de código, *snippets* e refatoração de código. Ele também é customizável, fazendo com que os usuários possam mudar o tema do editor, teclas de atalho, preferências e adicionar extensões (o PlatformIO IDE é uma extensão do VSCode). É um *software* livre e de código aberto (BRIGHT, 2015).

5.3 COMUNICAÇÃO E INTERFACES

5.3.1 Plataforma Node-Red

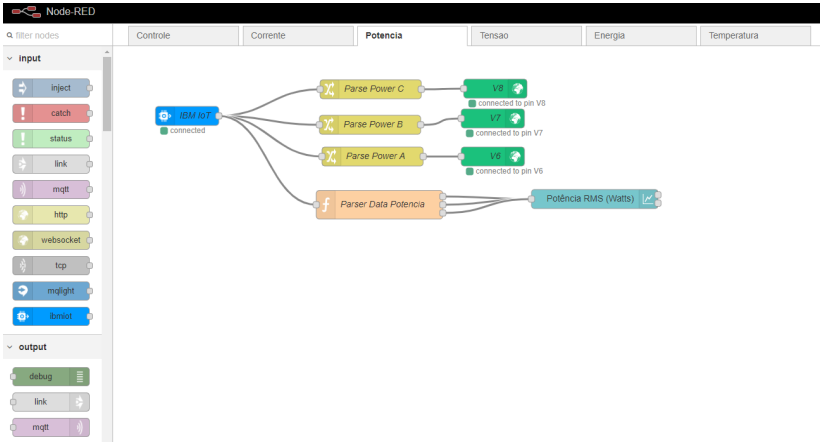
Node-Red é uma ferramenta de programação para conectar dispositivos de *hardware*, APIs, e serviços *on-line* de maneiras diferentes e interessantes. Fornece um editor baseado em navegador *web* que deixa o processo fácil e tem um vasto diretório de nós que podem ser utilizados com um simples clique (NODERED.ORG, 2013).

A ferramenta é construída em Node.js que a faz ideal para rodar em *hardwares* de baixo custo e também na nuvem. Ainda funções em JavaScript podem ser criadas dentro do editor utilizando um editor de texto (NODERED.ORG, 2013).

A ferramenta Node-Red, neste projeto, é utilizada para conectar a plataforma em nuvem com os dispositivos de hardware, a comunicação em sua maioria é feita pelo protocolo MQTT, a ferramenta também processa as informações recebidas, separando-as em seus respectivos gráficos relacionados as medidas adquiridas por cada hardware.

A Figura 20 apresenta exemplos de nós da ferramenta Node-Red, estes são os nós mais utilizados no projeto.

Figura 20 – Exemplo Node-Red.



Fonte: Do Autor

5.3.2 Aplicatio Blynk

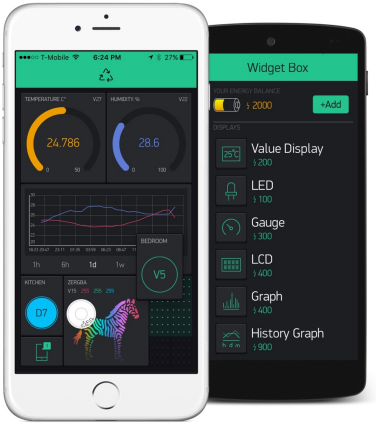
Blynk é uma plataforma desenvolvida para iOS e Android para controlar Arduino, Raspberry Pi e outros dispositivos através da Internet de uma forma extremamente fácil.

Consiste em um painel digital onde é possível construir uma interface gráfica para projetos, simplesmente arrastando e soltando *widgets*, sem complicações com plataformas mais complexas.

Ele foi um projeto desenvolvido pela Massachusetts Institute of Technology (MIT), com a finalidade de contribuir para o ensino e a aderência de mais pessoas para o conceito de IoT, aonde qualquer pessoa com um mínimo de noção de programação, consegue enviar e apresentar dados para o aplicativo.

Pela Figura 21, pode ser demonstrada sua interface. No lado direito da imagem, estão algumas opções de *widgets*, assim como no lado esquerdo da imagem, os *widgets* já estão em funcionamento. No exemplo tem-se dois medidores, um gráfico, dois botões e um painel RGB.

Figura 21 – Interface Blynk.



Fonte: (LISBOA; ROVERE, 2016)

O Aplicativo não está vinculado a algum dispositivo ou *shield* específico. Em vez disso, ele está apoiando o *hardware* de sua escolha. Se um Arduino ou Raspberry Pi está ligado à Internet através de Wi-Fi, Ethernet ou o chip ESP8266, o Blynk poderá ir on-line e expor todas

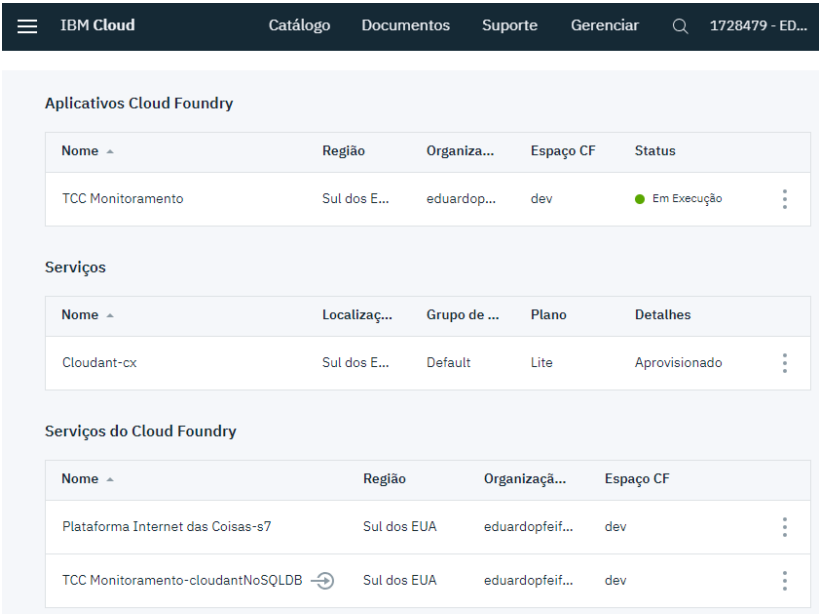
as informações importantes processadas que o projetista desejar.

5.3.3 IBM Cloud

IBM Cloud é uma oferta de nuvem da IBM que permite organizações e desenvolvedores criarem, implementarem e gerenciarem aplicativos de maneira fácil e rápida. Ele é composto por um painel apresentado na Figura 22 de onde é possível acessar todos seus serviços e plataformas, além de um catálogo onde é possível configurar novos serviços e criar novos aplicativos. Dentre os serviços encontrados no Cloud existem os de infraestrutura, cognitivos e de internet das coisas.

Este serviço de nuvem, utilizado juntamente com a plataforma Node-Red, foi base para este trabalho para a realização da comunicação entre todos os dispositivos e a interface Web (Node-Red) .

Figura 22 – IBM Cloud Painel.



Fonte: Do Autor

Similares ao IBM Cloud existem por exemplo o Amazon AWS e o Microsoft Azure. Provavelmente estes outros serviços também seriam

suficientes para a realização do mesmo projeto. A escolha do Cloud foi feita pela fácil integração com a ferramenta de desenvolvimento Node-RED já fornecida dentro do serviço da IBM.

5.4 BIBLIOTECAS UTILIZADAS

5.4.1 EmonLib

A EmonLib é uma biblioteca de monitoramento de eletricidade com comandos de cálculo e monitoramento de grandezas elétricas como tensão, corrente e potência, fornecida gratuitamente pela comunidade OpenEnergyMonitor (HUDSON, 2016). Esta biblioteca possui aplicações estritamente relacionadas com o desenvolvimento deste projeto, sua utilização é de extrema importância e facilita grande parte da programação em C do microcontrolador.

A biblioteca utilizada no código do Apêndice "A" foi desenvolvida para as placas Arduino que possuem uma tensão de entrada nos conversores A/D entre 0 e 5 V e uma resolução de 10 bits, portanto foi preciso realizar testes práticos para adaptar a biblioteca para a configuração do ESP32 que possui seu conversor A/D entre 0 e 3,3V e resolução de 12 bits.

As funções de cálculo contidas dentro da Emonlib utilizam constantes de entrada que podem ser alteradas para calibração, ou seja, esses valores deverão ser alterados de acordo com a especificação da medição a ser realizada.

As constantes adotadas foram práticas, ou seja, foram modificadas até os valores do monitor serial da IDE e dos instrumentos de medição entrarem em paridade.

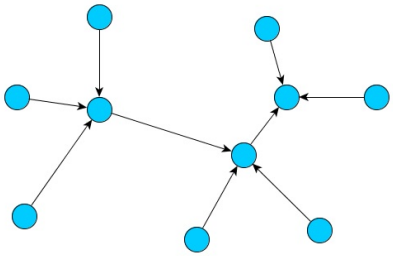
5.4.2 painlessMesh

A painlessMesh é uma biblioteca de código aberto que cuida dos detalhes da criação de uma rede *mesh* simples usando o hardware ESP8266 e ESP32. Assim como a biblioteca supracitada EmonLib, a painlessMesh é fundamental para a realização deste trabalho. Seu objetivo é permitir que o programador trabalhe com uma rede *mesh* sem ter que se preocupar sobre como a rede é estruturada ou gerenciada.

A biblioteca foi construída para ser uma verdadeira rede *ad-*

hoc [1], o que significa que não é necessário planejamento, controlador central ou roteador. Qualquer sistema de um ou mais nós se auto-organizará em uma malha totalmente funcional, tendo como característica uma topologia em estrela (Figura 23), evitando caminhos circulares.

Figura 23 – Topologia em estrela



Fonte: (MESSIAS, 2009)

A painlessMesh usa objetos JSON para todas as suas mensagens, tornando o código e as mensagens legíveis, facilitando a integração da mesma com front-ends javascript e aplicativos web.

Também vale ressaltar que a painlessMesh é projetada a partir das bibliotecas SDK (do inglês *Software Development Kit*) do ESP8266 e ESP32, nativas do fabricante dos microcontroladores e disponíveis através do Arduíno IDE.

Lista de recursos da biblioteca:

- Mensagens baseadas em JSON. Eles podem ser usados diretamente no código javascript;
- Instalação bastante fácil;
- Utilizável no ambiente Arduino;
- Implementado como um protocolo de camada 3. Sem conexão e sem confirmação;
- Totalmente assíncrono. Não há necessidade de pesquisar novas mensagens;

[1] redes ad-hoc são um tipo de rede que não possui um nó ou terminal especial, geralmente designado como ponto de acesso.

- Sincronização de tempo precisa. Todos os nós compartilham o mesmo *clock* com uma precisão menor que 10 ms;
- Todos os nós conhecem a topologia completa instantaneamente. Atualizada a cada 3 segundos;
- Malha autoconfigura-se. Qualquer nó pode ser desconectado em qualquer momento. Qualquer novo nó é integrado automaticamente;
- Todos os nós monitoram continuamente seus pares conectados diretamente para verificar se estão ativos;
- Mensagens individuais endereçadas e transmitidas são possíveis;
- Loops de rede são ativamente evitados.

6 SISTEMAS DESENVOLVIDOS

Neste capítulo será apresentado o detalhamento do sistema proposto com o intuito de realizar a medição do consumo de energia dos prédios da UFSC, Campus Araranguá.

6.1 DESCRIÇÃO

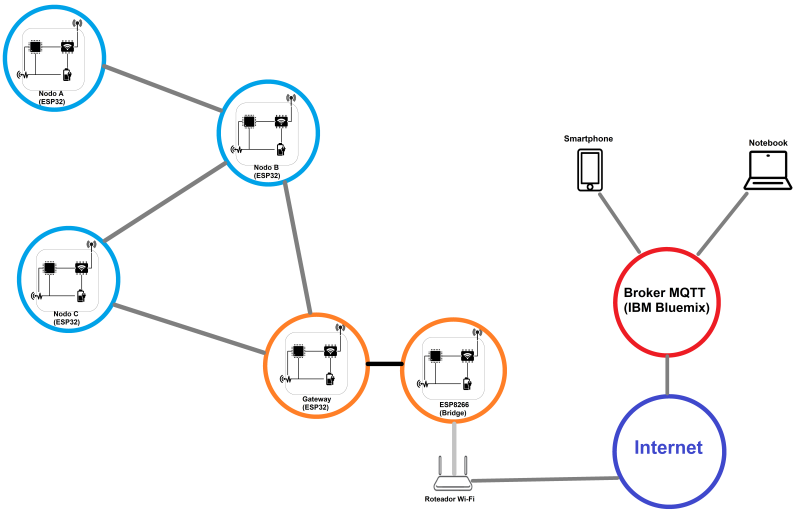
O sistema embarcado proposto engloba as tecnologias detalhadas nos capítulos anteriores. Com o objetivo de medir o consumo do ambiente em que o sistema está inserido, foi utilizado um sensor de corrente, tensão e temperatura. Mais precisamente, o sensor de tensão é responsável por determinar qual a tensão RMS da rede elétrica, o sensor de corrente a corrente RMS e o sensor de temperatura a temperatura da região onde se encontra os disjuntores da rede. Os sensores de tensão e corrente foram baseados no mesmo circuito proposto pelo projeto OpenEnergyMonitor (HUDSON, 2016).

Os sensores descritos são ligados a um microcontrolador com capacidade de realizar a aquisição dos dados através dos conversores Analógicos-Digital para os sensores de tensão e corrente, para o sensor de temperatura foi utilizado o protocolo 1-Wire. O microcontrolador utilizado é o ESP32, mais em específico a placa NodeMCU-32S, já detalhado anteriormente.

Com a capacidade de realizar uma rede de sensores se fio, o microcontrolador estabelece uma comunicação entre os outros microcontroladores, formando uma rede de comunicação em malha, assim, garantindo a expansão da rede para os microcontroladores que não se encontram sob a cobertura da rede *Wi-Fi* da Universidade.

Os dados já capturados pelos microcontroladores (chamados de Nodos) são repassados até um Nodo raiz da rede, chamado de *Gateway*. O *Gateway* é responsável por receber os dados de todos os Nodos sensores e repassa-los para um segundo microcontrolador, o ESP8266, que, a partir da comunicação serial recebe os dados e envia-os ao *Broker* MQTT (IBM Cloud) pela rede Wi-Fi da UFSC. Os dados já recebidos pelos *Broker* MQTT são tratados e apresentados de forma gráfica em uma página *WEB* desenvolvida a partir da ferramenta Node-RED e também disponibilizada no aplicativo Blynk. A Figura 24 representa a configuração acima detalhada do sistema.

Figura 24 – Representação do sistema descrito.



Fonte: Do Autor

6.2 REQUISITOS DO SISTEMA

Nesta seção será descrito os requisitos funcionais e não funcionais do sistema proposto.

6.2.1 Requisitos Funcionais

- Aquisição da corrente do dispositivo/local de medição;
- Aquisição da tensão do dispositivo/local de medição;
- Aquisição da temperatura do disjuntor;
- Cálculo da potência do dispositivo/local;
- Cálculo da energia consumia;
- Cálculo da tensão média;
- Transmissão dos dados obedecendo a topologia mesh;

- Acesso aos dados on-line por meio de uma página WEB e um aplicativo Mobile;
- Apresentar gráficos e indicadores dos dados coletados e calculados.

6.2.2 Requisitos não Funcionais

- Realizar as leituras de corrente e tensão a cada 1 segundo;
- Realiza a leitura de temperatura a cada 5 segundos;
- Envio dos dados dos nodos sensores para o *Gateway* a cada 5 segundos;
- Envio imediato dos dados do Gateway para o Bridge;
- Envio imediato dos dados do *Bridge* para o *Broker* MQTT;
- Apresentar uma tela de *login* (usuário e senha) para acesso a página *WEB*;
- Apresentar o histórico dos dados nos gráficos com intervalo de pelo menos 30 minutos;
- Utilizar uma bateria para alimentar o sistema;

6.3 SISTEMA DE AQUISIÇÃO DE DADOS

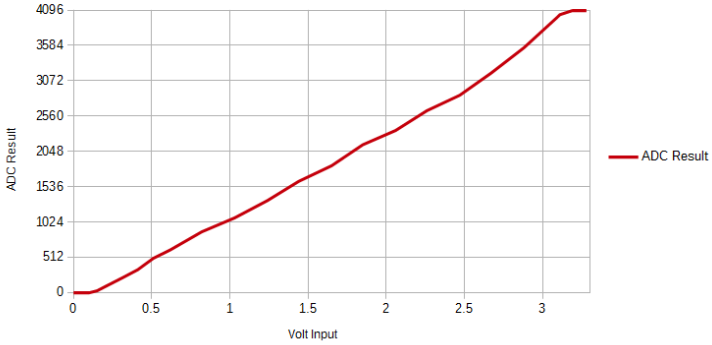
Com o objetivo de sentir o ambiente em que o sistema está inserido, foi utilizado um sensor de corrente, tensão e temperatura. Mais precisamente, o sensor de tensão é responsável por determinar qual a tensão RMS da rede elétrica, o sensor de corrente a corrente RMS e o sensor de temperatura a temperatura da região onde se encontra os disjuntores da rede. Os sensores de tensão e corrente foram baseados no mesmo circuito proposto pelo projeto OpenEnergyMonitor (HUDSON, 2016).

6.3.1 Conversor A/D do Microcontrolador

A técnica utilizada para leitura de um sinal analógico pelo ESP32 é a conversão analógica digital (conversor A/D). Com essa técnica, pode-se quantificar o sinal analógico presente em um pino de entrada do microcontrolador para um valor digital conforme a quantidade de bits da sua resolução.

A resolução de um conversor A/D é dada pela tensão de referência (V_{Ref}) dividida pelo número de bits do conversor (2^n). O microcontrolador ESP32 possui um conversor de 12 bits de resolução e sua V_{Ref} pode variar de 0 V até o valor de VCC (3,3 V) como mostrado na Figura 25. O cálculo da sua resolução está representado na Equação 6.1.

Figura 25 – Linearidade do Conversor Analógico-Digital.



Fonte: (LISBOA; ROVERE, 2016)

$$Resolução = \frac{V_{Ref}}{2^n} \tag{6.1}$$

$$Resolucao_{ESP32} = \frac{3,3\ V}{2^{12}} = \frac{3,3\ V}{4096} = 80566\ \mu V$$

6.3.2 Taxa de Amostragem e Teorema de Nyquist–Shannon

Quando trata-se de aquisição de sinais analógicos com formas senoidais, conversores A/D e resolução, é indispensável a análise da taxa de amostragem que o conversor fornece em relação à frequência do sinal lido.

A taxa de amostragem é a quantidade de amostras de um sinal analógico coletadas em uma determinada unidade de tempo para a conversão de um sinal digital. Sendo uma frequência, é comumente medida em Hertz (Hz), porém quando tratada por conversores A/D de microcontroladores, é medida em Amostras por Segundo (SPS) do inglês *Samples per Second*. De acordo com o *datasheet* (SYSTEMS, 2018), o microcontrolador ESP32 possui um conversor A/D do tipo SAR (*Successive Approximation Register*) com taxa de amostragem de 2 Mega sps.

O Teorema de Nyquist–Shannon diz que, um sinal analógico, limitado em banda, que foi amostrado, pode ser perfeitamente recu-

perado a partir de uma sequência infinita de amostras, se a taxa de amostragem for maior que duas vezes a maior frequência do sinal original. Caso contrário, não é possível reconstituir o sinal.

Observando que a frequência da rede elétrica no Brasil é de 60 Hertz, a taxa mínima de amostragem neste caso é de 121 Hertz, ou melhor dizendo, 121 sps.

Tendo o conhecimento da frequência do sinal analógico e da capacidade de amostragem do conversor A/D do microcontrolador, faz-se necessário deste ponto em diante, a garantia do algoritmo programado no microcontrolador respeitar o teorema de Nyquist–Shannon.

O Algoritmo utilizado neste trabalho é o da biblioteca EmonLib, que permite o desenvolvedor escolher o número de amostras desejadas e um tempo máximo para a aquisição (do inglês time-out), sendo escolhido um número de 2000 amostras para se ter uma boa amostragem e um time-out de 2 segundos, como mostrado na linha 78 no Apêndice A (Código fonte dos nodos).

6.3.3 Aquisição de Corrente

6.3.3.1 Transformadores de Corrente

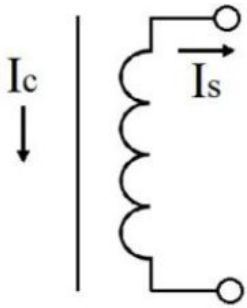
Um transformador de corrente (TC) é composto por um conjunto de espiras ao redor de um condutor do qual se deseja medir a corrente. O TC tem uma corrente alternada induzida (Is) em seus polos, que é diretamente proporcional a corrente alternada (Ic) que percorre o condutor (Figura 26). A Is no TC é, também, proporcional ao número de espiras de sua bobina.

Os TCs, ou sensores de corrente, são apresentados em 2 tipos: os *split-core* (núcleo dividido) e os *solid-core* (núcleo sólido). Neste trabalho foi utilizado o *Split-core Current Transformer* 013 (SCT-013), que possui uma bobina interna em sua estrutura, como podemos ver na Figura 27.

6.3.3.2 Os Modelos de SCT-013

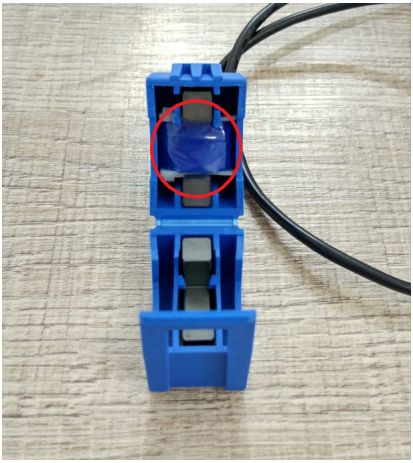
Com base nos princípios de funcionamento de um TC já discutidos acima, foram criados diferentes tipos de sensores não invasivos, como pode ser observado no *datasheet* do fabricante (YHDC, 2010, Beigaozhuang Village, Haigang district, China). A Figura 28 apresenta

Figura 26 – Esquema para exemplificar o SCT-013.



Fonte: (DEMETRAS, 2017)

Figura 27 – Imagem do SCT-013-000 e, em destaque com círculo vermelho, a bobina.



Fonte: Do Autor

os modelos SCT projetados para atender os mais diversos cenários, de forma que não exista um "melhor", mas sim o mais recomendado para cada aplicação. A vantagem de se utilizar um sensor de corrente não invasivo para este estudo é a de não precisar modificar o circuito original para realizar a medição da corrente. No caso de um sensor invaso, seria necessário abrir o circuito, produzindo alterações indesejáveis no mesmo.

Figura 28 – Diferentes modelos do TC não invasivo.

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

Fonte: (YHDC, 2010)

As principais diferenças entre os modelos são a corrente eficaz máxima a ser medida (*Input current*) e o tipo de saída do sensor (*Output type*).

Na figura 28, é possível observar que somente o modelo SCT-013-000 apresenta uma variação de corrente em sua saída (0-50 mA), já os outros modelos apresentam uma variação de tensão (0-1 Volt), onde, por meio destas variações, é possível mensurar a corrente elétrica.

Para determinar a taxa variação tanto de corrente quanto de tensão, é preciso dividir o valor máximo da saída pelo valor máximo a ser medido, como pode ser observado no exemplo do modelo SCT-013-000 a seguir:

$$TaxaDeVariacaoDeCorrente = \frac{0,05}{100} = 0,5\ mA$$

A partir desse cálculo, pode-se concluir que a cada um Ampere a mais ou a menos, sua saída será de 0,5 mA a mais ou a menos.

Neste trabalho o modelo de TC utilizado foi o SCT-013-000, representado na Figura 29. Ele pode medir valores de 0 até 100 A AC, mostrando-se, assim, ser a melhor opção dentre os modelos supracitados (Figura 28) por conta de sua corrente de entrada possibilitar medições de alta potência, como em disjuntores. Em sua saída, por sua vez,

apresenta valores entre 0 a 50 mA, proporcionais ao valor de corrente percorrido no condutor principal.

Figura 29 – Transformador de corrente YHDC SCT-013-000.

Fonte: (YHDC, 2018)

6.3.3.3 Corrente Gerada Pelo Sensor

O sensor é construído para medir corrente alternada máxima de 100 A AC. Esse valor de 100 A é o valor RMS, que também é chamado de valor eficaz. O valor RMS é igual ao máximo valor que a corrente pode alcançar (corrente de pico) dividido pela raiz quadrada de dois.

Então, temos que a corrente de pico máxima medida é de:

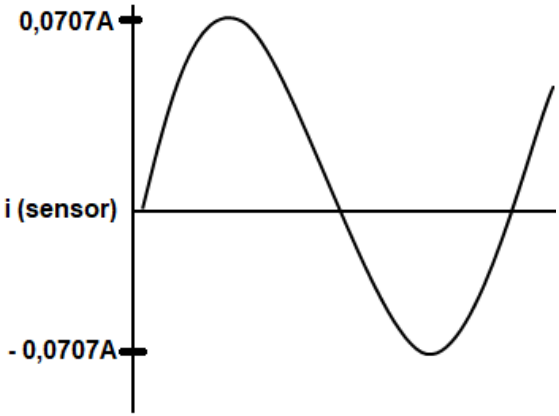
$$i(medido) = \sqrt{2} \times i(RMS) = 1,414 \times 100\ A = 141,4\ A$$

Sabendo-se que para uma corrente de 100 A no primário, ele produz 50 mA no secundário, basta reproduzir esses valores na Equação 6.2 de relação de transformação. O resultado será:

$$\frac{N_1}{N_2} = \frac{I_2}{I_1} \tag{6.2}$$

- I_1 = corrente no primário (corrente a ser medida);
- I_2 = corrente no secundário;
- N_1 = número de votas do primário (no caso deste sensor, N_1 será igual a 1);

Figura 30 – Sinal na saída do SCT-013 para um valor de corrente eficaz de 100 A.



Fonte: Do Autor

- N_2 = número de voltas do secundário.

$$N_2 = 2000$$

A corrente na saída do sensor é inversamente proporcional ao número de voltas (2000):

$$i(sensor) = \frac{i(medido)}{NumeroDeEspiras} = \frac{141,4 \text{ A}}{2000} = 0,0707 \text{ A}$$

Teremos na saída do sensor o sinal da corrente induzida semelhante ao da Figura30 a seguir:

6.3.3.4 Transformando Corrente em Tensão

Como mencionado anteriormente, conhecer as configurações do conversor analógico-digital possibilita a otimização e confiabilidade do sinal coletado. Portanto, são considerados os mesmos limites de *hardware* do conversor A/D da captura da tensão. Entendendo que o microcontrolador ESP32 realiza em seus pinos de entrada analógica a leitura de níveis de tensão entre 0-3,3 V, é necessário converter o sinal de corrente do SCT-013 para um valor de tensão que seja legível para o mesmo.

Para realizar a conversão do sinal de corrente vindo do TC a um sinal de tensão, será utilizado um resistor de carga (R_{carga}) posicionado entre os terminais do TC. O cálculo do resistor de carga é representado pela Equação 6.3.

$$R_{carga} = \frac{V_{ref} \times N_2}{\sqrt{2} \times I_{PrimarioMaxima}} \tag{6.3}$$

Onde: V_{Ref} - Tensão de referência do ESP32 ($V_{Ref}/2$).

Como o ESP32 opera em 3,3 V: V_{Ref} será de 1,65 Volts. Assim, o resistor de carga será:

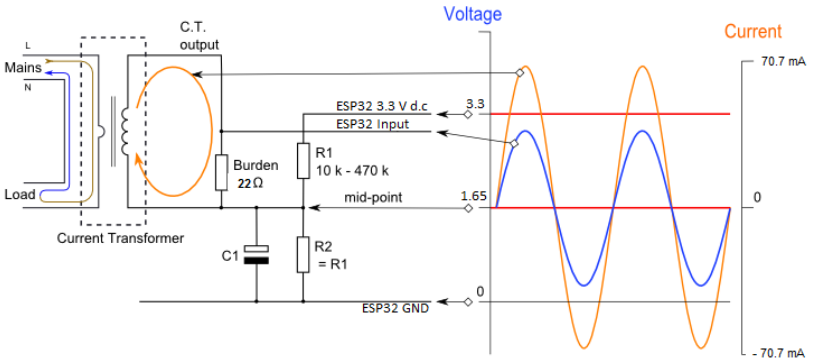
$$R_{Carga} = \frac{1,65 \times 2000}{\sqrt{2} \times 100} = 23,33\ \Omega$$

Entendendo que 23,33 Ω não é um valor de resistência comum, o valor comercial mais próximo de 23,33 Ω é 22 Ω. Quanto mais próximo do ideal for o valor do resistor de carga, melhor será a precisão (HUDSON, 2016).

Feito a conversão do sinal em níveis de corrente para tensão, o objetivo nesta etapa seguinte é obter o valor do ciclo positivo e negativo da forma de onda vindo do TC. Utilizando um circuito divisor resistivo foi possível reduzir a amplitude da onda desejada. A Figura 31 exemplifica o circuito desenvolvido para o sensor de corrente.

O divisor resistivo é composto por 2 resistores de valores iguais, variando de 10 kΩ a 470 kΩ, sendo que quanto maior o valor da resistência, menor será o consumo de energia e interferência no sinal analisado. A tensão aplicada na entrada do divisor resistivo é a tensão de referência do ESP32 (3,3 Volts). Já os terminais do TC são conectados

Figura 31 – Diagrama do circuito divisor resistivo.



Fonte: Adaptado (HUDSON, 2016)

na saída do divisor resistivo e na entrada do A/D. A função do divisor neste caso é fornecer um *offset* de 1,65 V, elevando o eixo referencial e impedindo que níveis de tensão fiquem abaixo de 0 V. Entendendo que a corrente máxima de pico do TC é de 0,0707 A, a tensão entre os terminais do resistor de carga (22 Ω), de acordo com a Lei de Ohm, temos:

$$V = R \times I \tag{6.4}$$

$$V_{Carga} = 22\Omega \times \pm 0,0707 \text{ A} = \pm 1,5554 \text{ V}$$

Ou seja, o sinal na entrada do conversor A/D ($V_{A/D}$) com o ajuste de offset irá excursionar entre:

$$V_{A/D} = -1,5554 \text{ V} + 1,65 \text{ V} = 0,0946 \text{ V}$$

e

$$V_{A/D} = +1,5554 \text{ V} + 1,65 \text{ V} = 3,2054 \text{ V}$$

obedecendo portanto, os limites do conversor A/D (0-3,3 V).

Vale ressaltar uma questão muito importante em relação ao sinal de referência (3,3 V) utilizado para determinar o *offset*. Caso o mesmo não permaneça constante durante a aquisição do sinal, valores não realísticos poderão ser capturados, resultando em um cálculo errôneo da

corrente/tensão e sucessivamente da potência e energia. Portanto, a implementação de um circuito mais robusto como um regulador de tensão exclusivo para o sinal de referência, pode ser necessário em casos de variação da tensão vinda da fonte de alimentação do microcontrolador.

O código para a aquisição dos dados bem como a biblioteca utilizada (EmonLib) pode ser encontrada no Apêndice A ou consultando o projeto OpenEnergyMonitor (LEA TRYSTAN. HUDSON, 2016) já mencionado.

6.3.4 Aquisição de Tensão

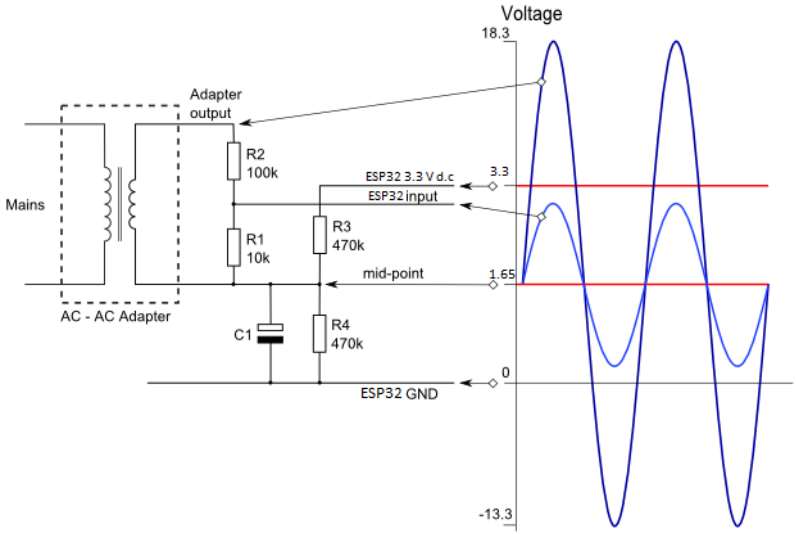
O circuito responsável pela aquisição está ilustrado na Figura 32, que consiste em um transformador de tensão alternada da rede elétrica para alternada (e reduzida) para o sensor A/D. O sinal de saída do adaptador de tensão AC (*Alternating Current*) é proporcional à tensão de entrada e possui uma forma de onda quase senoidal.

Neste caso, foi utilizado um transformador que possui como saída uma tensão nominal de 9 Volts RMS, pico positivo de 12,7 V e negativo de -12,7 V. No entanto, devido à má regulação desse tipo de transformador, quando o mesmo não possui carga (como neste caso), a saída é geralmente entre 10-12 V (RMS), fornecendo uma tensão de pico de 14-17 V. Nas medições feitas para este trabalho, a tensão RMS para o transformador com 9 V RMS utilizado foi de 10 V RMS.

O microcontrolador precisa converter a saída do transformador para uma forma de onda que tenha um pico positivo menor que 3,3 V (tensão máxima do conversor Analógico-digital) e um pico negativo maior que 0 V como já descrito na aquisição da corrente.

A forma de onda pode ser reduzida usando um divisor de tensão conectado através dos terminais do adaptador e um *offset* (polarização) deve ser adicionado usando uma fonte de tensão criada por outro divisor de tensão conectado na fonte de alimentação do microcontrolador, da mesma forma que foi adicionado uma polarização para o circuito de detecção atual. O propósito de se adicionar um *offset* no sinal é por conta da natureza da onda ter pico de tensão positivo e negativo. O sensor A/D do microcontrolador não lida com valores menores que 0 V, então adicionando um *offset* é possível corrigir a faixa de tensão lida por ele. A Figura 32 apresenta o diagrama do circuito e as formas de onda de tensão.

Figura 32 – Diagrama AC-AC.



Fonte: Adaptado (HUDSON, 2016)

Os resistores R1 e R2 formam um divisor de tensão que diminui a tensão AC do adaptador de energia. Os resistores R3 e R4 fornecem o viés de tensão. O capacitor C1 fornece um caminho de baixa impedância ao terra para o sinal AC.

R1 e R2 precisam ser escolhidos para fornecer uma saída de tensão de pico de aproximadamente 1 V. Para um transformador AC-AC com uma saída de 9 V RMS, uma combinação de resistor de 10 kΩ para R1 e 100 kΩ para R2 foi adequado. A Equação 6.5 representa o cálculo feito.

$$TensaoPicoSaida = \frac{R1}{R1 + R2} * PicoInicialEntrada \tag{6.5}$$

$$TensaoPicoSaida = \frac{10\ \text{k}}{10\ \text{k} + 100\ \text{k}} * 14.14\ \text{V} = 1.28\ \text{V}$$

A polarização de tensão fornecida por R3 e R4 deve ser metade da tensão de alimentação do microcontrolador. Como tal, R3 e R4

precisam ser de igual resistência. Para a utilização de baterias, onde o baixo consumo de energia é importante, foi utilizado resistores de 470 kΩ para R3 e R4.

Com o microcontrolador funcionando a 3,3 V, a forma de onda resultante terá um pico positivo de:

$$V_{A/D} = -1,28 \text{ V} + 1,65 \text{ V} = 0,37 \text{ V}$$

e

$$V_{A/D} = +1,28 \text{ V} + 1,65 \text{ V} = 2,93 \text{ V}$$

satisfazendo os requisitos de tensão de entrada analógica do microcontrolador. Isso também deixa uma faixa de tensão livre para minimizar o risco de excesso ou baixa tensão.

6.3.5 Aquisição da Temperatura

O sensor DS18B20 (Maxim Integrated Products, 2018, 160 Rio Robles San Jose, CA 95134 ,USA) utilizado neste trabalho possui algumas características especiais e, mesmo tendo o custo um pouco acima da média, ainda é bastante acessível e de fácil utilização. Dentre formas de encapsulamento existentes, a utilizada para este estudo é o encapsulamento blindado com bulbo de aço inox e vedação dos fios que o torna a prova d'água (Figura 33).

Figura 33 – Sensor DS18B20.

Fonte: Do Autor

As principais características do DS18B20 são:

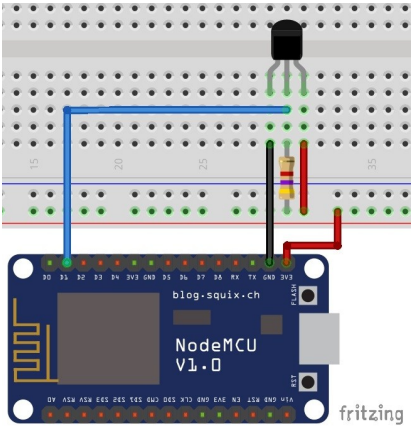
- Alimentação DC entre 3,0 V e 5,5 V;
- Mensuração de temperatura entre -55 °C e +125 °C;
- Não necessita de componentes externos;
- Resolução ajustável entre 9 e 12 bits;
- A prova de água, umidade e oxidação;
- Utiliza o Protocolo 1-Wire para leitura;
- Cada sensor tem um número serial.

O sensor DS18B20 possui dois tipos de funcionamento, o chamado "Normal", que é feito utilizando os três fios, sendo GND, Vcc, e o sinal para o microprocessador nesse caso para uso unitário, quando será utilizado apenas um sensor no circuito, o outro modo é o modo "Parasita" onde os fios de Vcc e do sinal são curto-circuitados. Nesse modo, pode-se utilizar a tecnologia 1-wire, onde é possível combinar até 127 sensores. Basicamente o Protocolo 1-Wire define algumas características que tornam possível utilizar vários sensores conectados a apenas um pino de leitura do Microcontrolador. Algumas características do Protocolo 1-Wire são:

- Definição de um *Serial Number* em cada sensor compatível, tornando possível;
- Identificar individualmente cada sensor conectado no mesmo pino de leitura (podemos chamar de barramento);
- Definição de um CRC (Cyclic Redundancy Check) para podermos tratar possíveis colisões no barramento (dois sensores enviando informação ao mesmo tempo);
- Definição de meios para leitura e controle dos sensores;
- Tipos de sensores DS18B20.

A conexão apresentada na Figura 34 é um método simples, amplamente utilizado com esse sensor. O pino GND é ligado ao GND do ESP32, o Vcc ao 3,3 V e o DATA ao GPIO15 que é utilizado no código para leitura dos dados, no exemplo da Figura 34 foi utilizado o pino D1. Para a captura dos dados foi utilizado duas bibliotecas, a OneWire para a comunicação do sensor e a DallasTemperature para o cálculo da temperatura.

Figura 34 – Esquema do sensor DS18B20.



Fonte: (SANTOS, 2016)

6.3.6 Calibração dos Sensores de Tensão e Corrente

Quando trata-se de calibração de algum aparelho de medição, é necessário levar em conta o erro que está implícito no processo. Existem três fontes principais de erro ao usar o circuito proposto pelo (LEA TRYSTAN. HUDSON, 2016) como instrumento de medição. Os transdutores de entrada que contempla o primeiro estágio na conversão e dimensionamento da quantidade a ser medida, o circuito de entrada que completa a tarefa de condicionar o sinal e o próprio conversor A/D. A seguir será apresentado a faixa de erro dos pontos mencionados acima.

- **Transformador de Corrente**. O TC utilizado (SCT-013-000) é linear dentro de 3% da faixa de corrente nominal. O datasheet (YHDC, 2010) não dá nenhum valor para a precisão da relação de voltas - como é uma questão de contagem, só podemos assumir que ela é precisa em um grau muito alto, provavelmente algumas voltas, ou uma pequena fração de 1%.

- **Transformador de Tensão**. O transformador de tensão utilizado neste trabalho tem sua tensão de saída sem carga medida em 10 Volts RMS e com uma precisão especificada de ± 3%. Esta tensão não deve ser confundida com a diferença da saída nominal de 9 Volts RMS que é especificada em carga máxima (ambos com entrada de 220 Volts).

- **Circuito de Entrada**. Existem duas partes principais no circuito de entrada, os componentes de escala e os componentes de polarização. Apenas os componentes de dimensionamento são de interesse. Embora os componentes de polarização possam afetar o intervalo de medição, eles não contribuem para nenhum erro de medição.

    1. Entradas atuais. O resistor de carga (burden), necessário para converter a saída de corrente do TC em uma tensão, é o único componente de escala. É um item padrão, com uma tolerância de fabricação de 1%.
    2. Tensão de entrada. A tensão de entrada é aplicada a um potencial divisor compreendendo dois resistores, cada um com uma tolerância de 1%. O pior caso nos dá um erro na relação de divisão de 1,83% (assumindo ambas as mudanças na mesma proporção devido a variações de temperatura e, portanto, sua relação não muda devido à temperatura).

- **Conversor A/D**. O conversor A/D é o estágio final do processo de entrada. Existem duas principais fontes de erro: a precisão do processo de conversão e a precisão da tensão de referência.

    Segundo dados da Espressif (SYSTEMS, 2018), o erro máximo de leitura considerando um sinal de até 2,450 mV é de ± 3,63 % e também levando em conta que os chips ESP32 podem apresentar uma diferença de ± 6% de um chip para outro nos resultados medidos.

    No caso da tensão de referencia existem duas fontes, a referência de intervalo de banda interna ou a tensão de alimentação analógica AVcc. Na placa utilizada neste trabalho a tensão de referencia usada no conversor A/D é a AVcc, portanto o erro é, naturalmente, totalmente dependente do regulador de tensão. A fonte de 3,3 V para o ESP32 vem de um regulador MCP170 e sua especificação de tensão de saída é tipicamente ± 0,4%. Considerando os ± 3,63 % do leitor A/D e os ± 0,4% da tensão de referência, somando temos um erro de ± 4,03%.

A Tabela 8 apresenta os valores mencionados a cima e também o valor total do erro máximo estimado no cálculo da potência. Neste caso não foi considerado a precisão do aparelho de medição utilizado para a calibração dos sensores.

Tabela 8 – Tabela de precisão estimada dos componentes.

Componentes	**Erro**
Transformador de Corrente	± 3%
Resistor de Carga	± 1%
Referência Analógica	± 3.63%
Total	± **7,63%**
Transformador de Tensão	± 3%
Divisor de Tensão	± 1.83%
Referência Analógica	± 3.63%
Total	± **8,46%**
Erro de Potência	± **16,09%**

Para o circuito considerado, nas piores condições possíveis e antes da calibração, uma medição de tensão ou corrente pode estar com erro em quase 9% e uma medição de potência (real ou aparente) pode estar com erro em cerca de 16%.

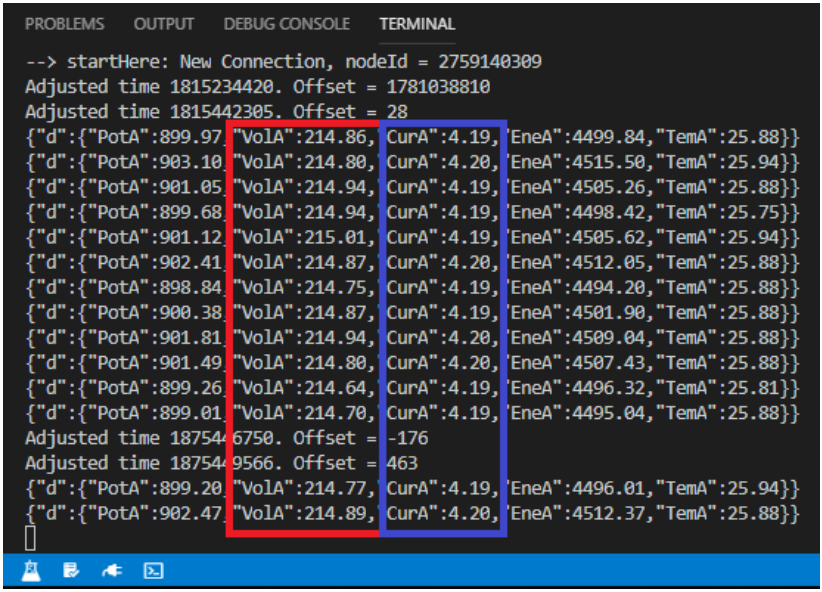
Para a calibração da tensão e da corrente foi utilizado um multímetro modelo Hikari-HM-2010 e um alicate amperímetro modelo Hikari HA-266 respectivamente. O amperímetro envolveu o mesmo mesmo fio do sensor SCT-013-000 e o multímetro ficou ligado em paralelo com a tomada. Com o microcontrolador já em funcionamento, os dados lidos foram enviados e analisados no monitor serial da IDE (Figura 36), os dados em vermelho representam a tensão lida e em azul a corrente. A Figura 35 exibe o protótipo com os sensores e medidores instalados.

A carga utilizada nesse teste foi um secador de cabelos com o valor de potência fornecida pelo fabricante de 1200 Watts.

Figura 35 – Protótipo sendo medido.

Fonte: Do Autor

Figura 36 – Dados do monitor serial.



Fonte: Do Autor

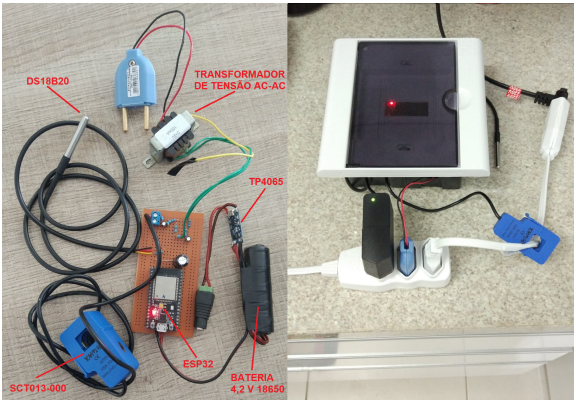
Com base na corrente e tensão apresentadas nos aparelhos de medição, a calibração foi sendo feita através do seguinte trecho de código do nodo:

```
1  emon1.voltage(39, 226, 1.7);
2  emon1.current(36, 90.9);
```

Em relação a calibração da tensão (linha 1), o número 39 representa o GPIO (do inglês *General Purpose Input/Output*) do microcontrolador onde se encontra o sinal de tensão a ser medido, o número 226 representa o valor de ajuste necessário para calibrar a tensão e o 1.7 é a mudança de fase da onda, do inglês (phase shift). Para a calibração não foi considerado o ajuste de fase.

Já para a calibração da corrente (linha 2), o número 36 representa o GPIO do sinal de corrente e o número 90.9 o ajuste da leitura da corrente, encontrado realizando a divisão do número de voltas do TC (2000) pelo valor da resistência do resistor de carga (22 Ω).

Figura 37 – Protótipo nodo sensor



Fonte: Do Autor

6.3.7 Protótipos

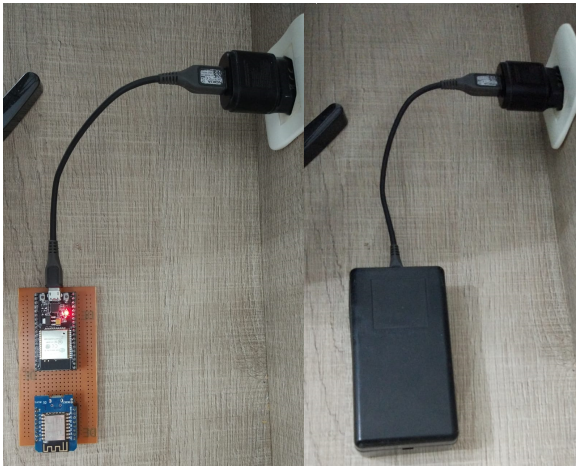
O protótipo serve como prova de conceito do trabalho, por ele podemos observar o desempenho da rede em ambiente real e documentar os problemas durante a implementação.

Foram desenvolvidos 2 tipos de protótipos, 3 do tipo Nodos/-Sensores (Figura 37) e 1 do tipo *Gateway/Bridge* (Figura 38). Cada nodo sensor possui:

- 1 Microcontrolador ESP32 (NodeMCU-32S);
- 1 Sensor de corrente (SCT-013-000);
- 1 Circuito para aquisição de tensão (transformador de tensão AC-220 V para AC-9 V);
- 1 Sensor de temperatura (DS18B20);
- 1 Módulo de bateria (Módulo carregador TP4056 + Bateria 4,2 V Modelo 18650).

Já o protótipo do *Gateway/Bridge* consiste de 2 microcontroladores, um ESP32 (Módulo modelo NodeMCU-32S) e um ESP8266 (Módulo modelo WeMos D1 Mini). O ESP32 é o *Gateway*, ou seja, o responsável por receber os dados dos nodos sensores e repassá-los para o ESP8266 por meio da comunicação serial (UART). O Microcontrolador ESP8266 é chamado de *Bridge* pois é o dispositivo responsável em

Figura 38 – Protótipo do nodo Gateway/Bridge.



Fonte: Do Autor

conectar-se a rede Wi-Fi local, fazer a comunicação entre ele e o *Broker* MQTT (IBM Bluemix), receber os dados vindo do *Gateway* e envia-los ao *Broker*.

Ambos os nodos possuem uma fonte de tensão de AC-DC de 5 V. A fonte de tensão tem como propósito o fornecimento de energia para a carga da bateria dos nodos sensores e como fonte primária de energia do nodo *Gateway/Bridge*.

6.3.8 Página *WEB*

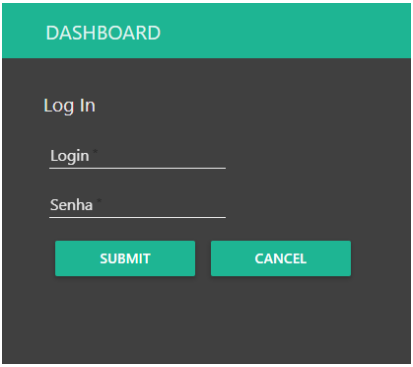
Como descrito na seção de requisitos funcionais, foi desenvolvido uma página *WEB* que disponibiliza gráficos de histórico da corrente, tensão, potência e energia como mostra a Figura 39.

Figura 39 – Gráficos indicadores de corrente, tensão, potência e energia.

Fonte: Do Autor

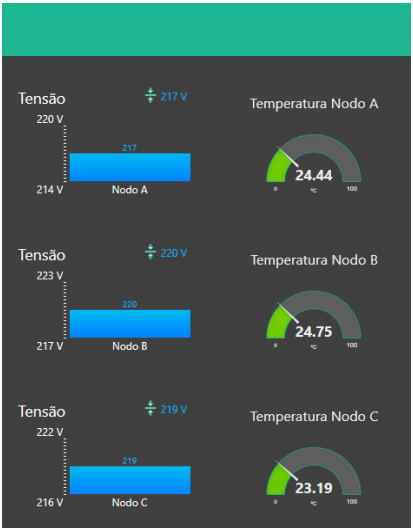
Também foi adicionado indicadores de tensão média e temperatura (Figura 41), bem como uma tela de *login* (usuário e senha) para que a visualização dos dados fique restrita somente a pessoas que possuir os dados de acesso (Figura 40).

Figura 40 – Página de Login.



Fonte: Do Autor

Figura 41 – Indicadores de tensão média e temperatura.

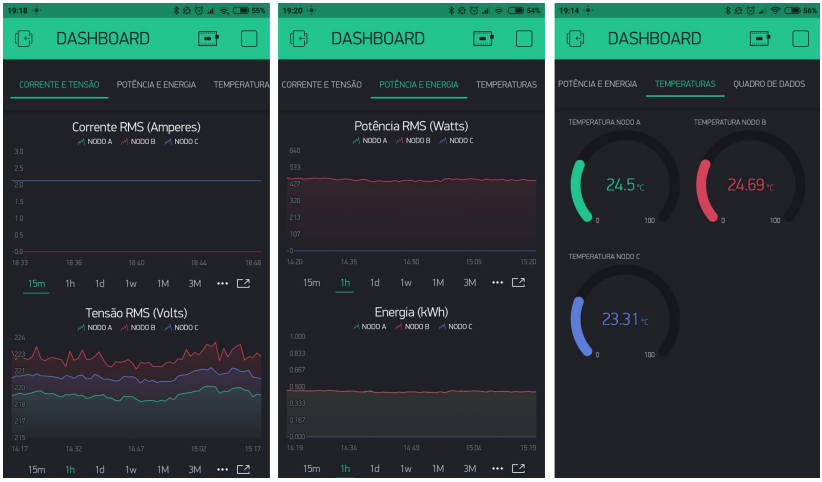


Fonte: Do Autor

6.3.9 Aplicativo

Aqui será exibido as telas desenvolvidas no aplicativo Blynk (Figura 42). Os mesmos gráficos e indicadores da página *WEB* foram criados.

Figura 42 – Indicadores de corrente, tensão, potência, energia e temperatura no aplicativo.



Fonte: Do Autor

7 TESTES E RESULTADOS

7.1 METODOLOGIA DOS TESTES

A fim de validar a proposta após a construção do sistema, foram realizados dois tipos de teste. O primeiro foi referente a utilização da rede *mesh* e suas possibilidades de configuração. O segundo, por sua vez, testou a instalação do sistema em uma residência por um determinado período para que fossem coletados dados reais de consumo.

7.1.1 Testes da Rede *Mesh*

Com o propósito de verificar se os dados coletados pelos nodos sensores percorrem a rede *mesh* até um nodo *Gateway* de forma independente da configuração da rede formada (Figuras 43 e 44), levando em conta variáveis como número de nodos, distância e intolerância a falhas, o sistema foi testado perante os seguintes cenários:

1. Um nodo e o Gateway ligados Figura 43.1;

2. Três nodos e o *Gateway* ligados (ambos com alcance entre todos) Figura 43.2;

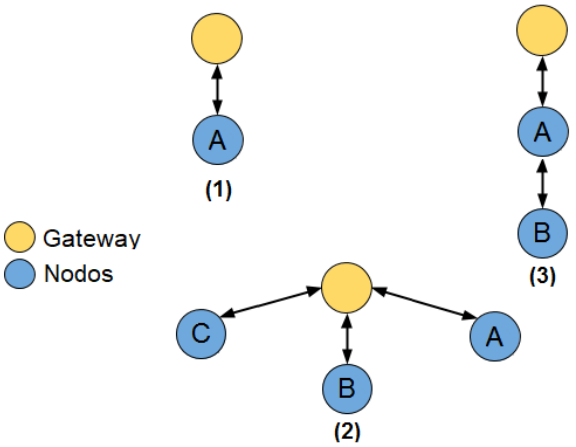
3. Dois nodos e o *Gateway* ligados, porém um dos nodos (A) tem somente o alcance do segundo nodo (B), e (B) tem alcance ao *Gateway* Figura 43.3;

4. Três nodos e o *Gateway* ligados, porém o nodo (A) e o nodo (B) tem somente o alcance de (C) e (C) ao *Gateway* Figura 44.4;

5. Três nodos e o Gateway ligados, porém o nodo (A) tem somente o alcance do nodo (B), e (B) tem somente o alcance do nodo (C) e (C) ao *Gateway* Figura 44.5;

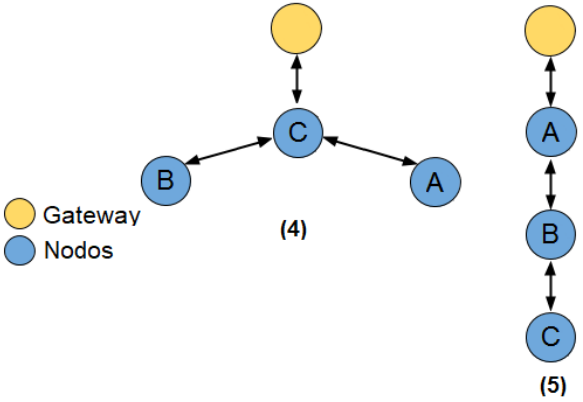
Durante a realização dos testes que possuem mais de um nodo ligado, foram aplicadas falhas propositais para avaliar o comportamento da rede em relação a comunicação e auto-organização. As falhas empregadas aos nodos foram do tipo: afastamento, desligamento e inserção de novos nodos (até 7 nodos, incluindo (A), (B), (C) e *Gateway*).

Figura 43 – Configurações da rede de testes tipo 1, 2 e 3.



Fonte: Do Autor

Figura 44 – Configurações da rede de testes tipo 4 e 5.



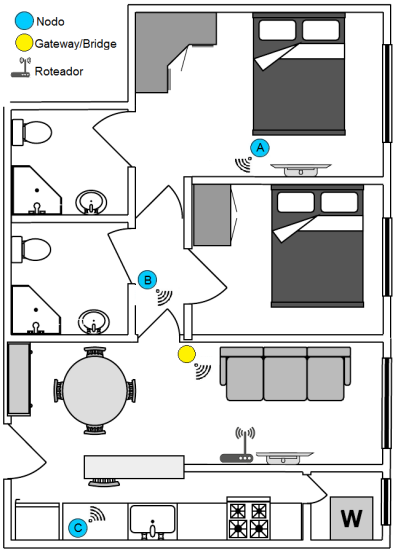
Fonte: Do Autor

7.1.2 Testes do Sistema

Tendo em vista as limitações de segurança e consentimento em relação a manipulação dos quadros de luz da Universidade, o sistema foi instalado em uma residência (apartamento, como representado na Figura 45) e os dados coletados foram provenientes de dispositivos em diferentes cômodos. Os pontos medidos foram os seguintes:

- Televisão LED 39' Samsung UN39FH5205 (nodo A);
- Notebook 14' Dell Inspiron 14Z (nodo B).
- Geladeira Frost Free Cuplex Electrolux DF42 (nodo C);

Figura 45 – Planta baixa da residência.



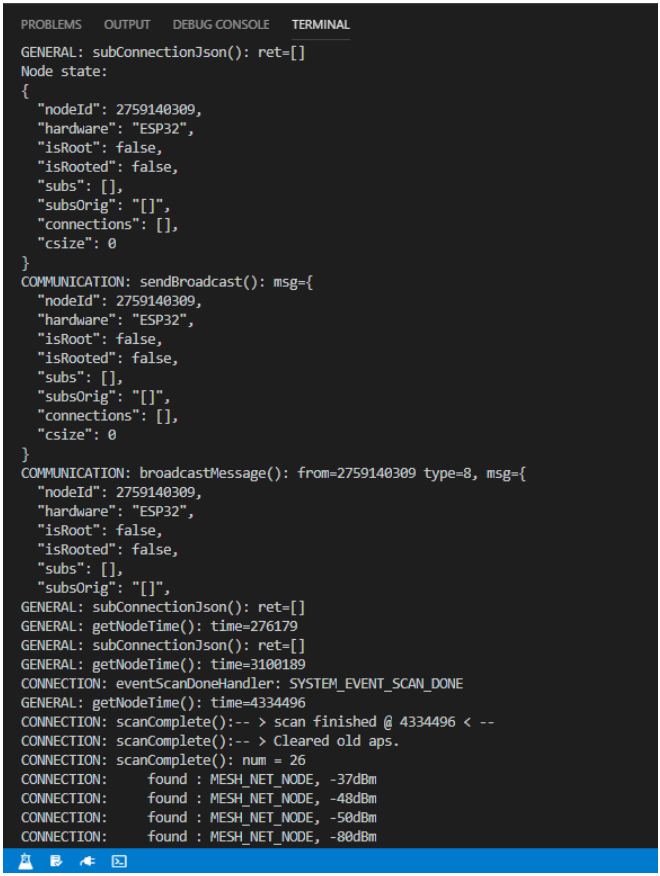
Fonte: Do Autor

7.2 RESULTADOS

7.2.1 Teste de Rede

Todos os testes de rede tiveram resultados correspondentes ao esperado pela biblioteca painlessMesh, ou seja, as mensagens foram entregues ao *Gateway* independente da topologia da malha, inclusive nas falhas propositais aplicadas.

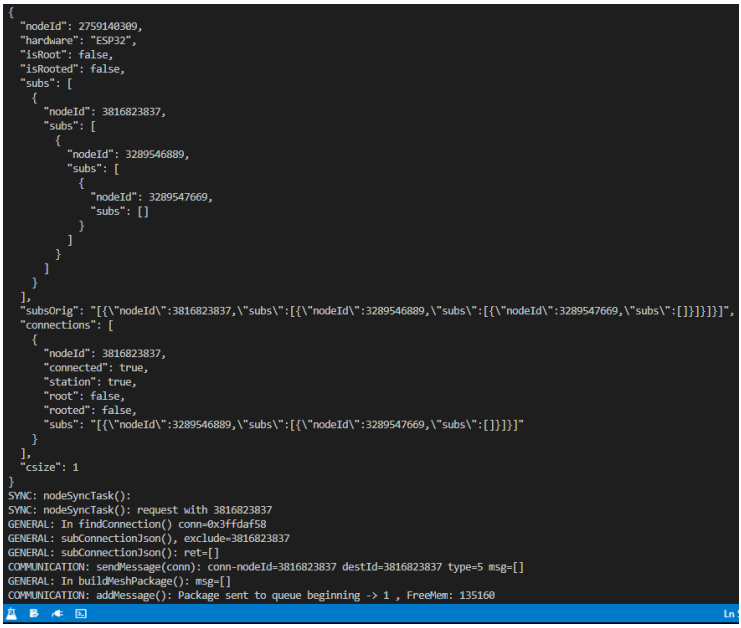
Figura 46 – *Debug* da malha de rede em formação 1.



Fonte: Do Autor

A complexidade da obtenção das informações relativas a formação da rede nas suas 5 diferentes configurações pode ser observada nas amostras de debug representadas nas figuras (46 e 47), representando o comportamento das trocas de mensagem entre os nodos. Os dados de debug exibiram informações a respeito dos nodos, tais como: estado, comunicação, potência do sinal, atualização do relógio e árvore de conexões.

Figura 47 – *Debug* da malha de rede em formação 2.

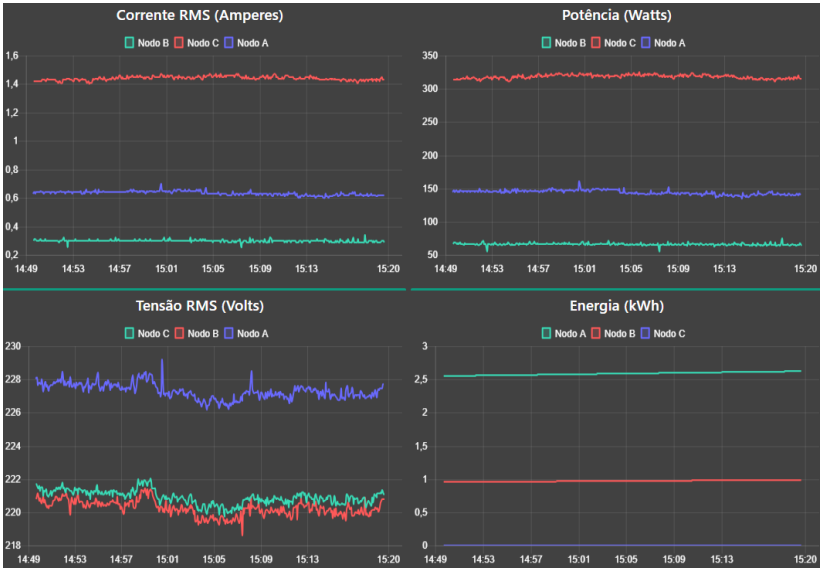


Fonte: Do Autor

7.2.2 Teste de Sistema

Referente ao teste do sistema, foram analisados os resultados dos gráficos mostrados na Figuras 48 e 49. De acordo com os aparelhos medidos, esperava-se uma potência consumida próxima da especificação do fabricante representada na Tabela 9.

Figura 48 – Gráficos dos dados coletados no teste do sistema.



Fonte: Do Autor

Tabela 9 – Potência nominal dos aparelhos medidos.

Aparelho	**Potência nominal**
Televisão LED 39’	150 Watts
Notebook 14’	65 Watts
Geladeira Frost Free Duplex	350 Watts

Fonte: Do Autor

Na Figura 48 o gráfico "Tensão RMS (Volts)"apresenta dados coerentes com a tensão esperada, com ressalvas do Nodo A que indica uma leve discrepância em relação aos outros. Isso se deve ao fato da bateria que alimenta o circuito não estar fornecendo uma tensão correta no momento do teste, ocasionando uma perda de referencia no sensor A/D e um erro no cálculo da tensão que se encontrava previamente calibrada.

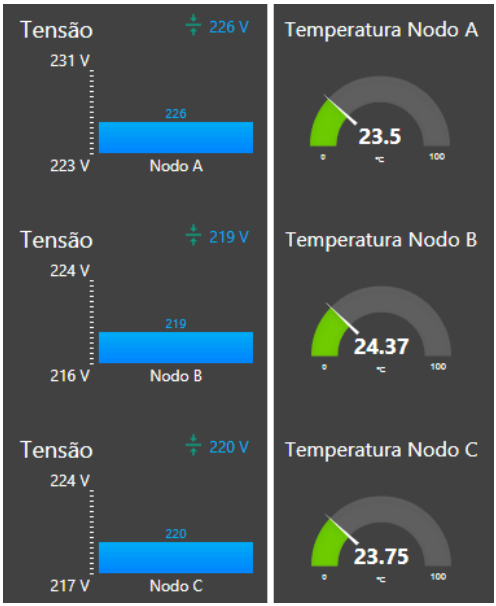
Já em relação ao gráfico "Energia (kWh)"é preciso mencionar que os dados de consumo não representam a realidade referente ao tempo do teste (30 minutos). Por se tratar de dados que não foram e não podem ser perdidos com o tempo, foram salvos durante o desenvolvimento e

testes do sistema, contemplando o consumo de outros dispositivos em momentos anteriores aos apresentados.

Os gráficos de Corrente e Potência confirmam que a medição dos dispositivos indicam valores muito próximos do esperado de acordo com a Tabela 9. Levando em conta o erro de leitura da tensão do Nodo A, entende-se que a potência apresentada também inclui um erro propagado, esperando-se um valor um pouco abaixo do apresentado.

Na Figura 49 que apresenta os indicadores de tensão média e temperatura, podemos comparar as informações do mesmo com as da Figura 48, principalmente em relação ao Nodo A indicando uma tensão média de 226 Volts. Nesses indicadores também é possível verificar os valores máximos e mínimos do experimento.

Figura 49 – Indicadores dos dados coletados no teste do sistema.



Fonte: Do Autor

8 CONSIDERAÇÕES FINAIS

8.1 CONCLUSÃO

Analisando o teste da rede sem fio em topologia *mesh*, esta se mostrou bastante eficiente e robusta, pois através dela os dados puderam ser trafegados independentemente da configuração da rede, sendo assim possível a instalação do sistema em qualquer local sem que exista uma pré configuração total da rede.

Já a partir do teste do sistema é importante levar em conta um detalhe antes da conclusão do mesmo. Foi percebido que o nodo responsável pela aquisição do consumo da Televisão (nodo A) não possuía cobertura *Wi-Fi* e não se conectava diretamente ao nodo *Gateway*. A causa desses detalhes pode-se deduzir pelo número de paredes, móveis/objetos que se encontram entre os dispositivos e também ao número elevado de redes *Wi-Fi* na região em torno da residência, ocasionando uma possível atenuação do sinal da frequência 2.4 GHz, a mesma frequência utilizada tanto no Roteador quanto nos nodos da rede *mesh*. Essa atenuação limita o alcance dos nodos, forçando um comportamento diferente na configuração da malha. O comportamento mencionado foi percebido que, mesmo pela não conexão direta entre o nodo A e o Gateway os dados estavam sendo exibidos nos gráficos, admitindo-se que a informação encontrou um caminho alternativo. Este caminho é nada menos que o nodo B realizando a função de nodo intermediário entre A e o *Gateway*, sendo verificado posteriormente desligando-se o nodo B e notando que a informação de ambos não foi mais exibida nos gráficos.

Portanto, os resultados obtidos através dos testes validam a hipótese deste trabalho, sendo possível observar que a implantação de um sistema de monitoramento proporciona uma visão mais detalhada do consumo de energia elétrica e a apresentação das características elétricas como corrente, tensão, potência e temperatura do disjuntor.

Não é possível deixar de levar em conta que a implantação do sistema de monitoramento na Universidade, apesar de não ter sido instalado na mesma para testes, pode gerar a inexistência da necessidade de uma rede em malha, caso a região monitorada já possua uma cobertura de Wi-Fi que abranja todos os nodos sensores. Um exemplo disso, é a implantação da rede em apenas uma sala, laboratório ou até mesmo um único andar, onde normalmente se tem uma cobertura total da região.

8.2 TRABALHOS FUTUROS

Uma sugestão para trabalhos futuros, é a substituição do módulo de rádio atual para uma rede LoRa. Essa rede utiliza uma banda de frequência de 915 MHz, que possibilita um alcance muito superior a rede 2.4 GHz utilizada no projeto, chegando a cobrir áreas acima de 3 km de raio, fazendo-se desnecessário a complexidade de utilização de algoritmos de roteamento para a formação de uma rede em malha.

Para que o sistema fique mais robusto em questões de informação, a possibilidade de envio de mensagens SMS informando falhas no fornecimento de energia através de um módulo GSM, também se torna uma melhoria importante quando o que está sendo medido é de suma importância e não pode sofrer interrupções.

Outro item importante que poderia ser integrado ao sistema desenvolvido, é a capacidade de análise de distúrbios da rede elétrica como, cintilação, cunha de tensão, desequilíbrio de tensão, elevação de tensão, afundamento de tensão, ruido e interferência eletromagnética. Com essa nova capacidade de detalhamento da rede, seria possível um diagnóstico mais preciso em caso de aparelhos eletrônicos que são sensíveis a algum desses tipos de distúrbios da rede elétrica.

REFERÊNCIAS

ADELLE, C.; PALLEMAERTS, M. *Climate Change and Energy Security in Europe Policy Integration and its Limits*. [S.l.: s.n.], 2009. ISBN 9789186107109.

ALLIANCE, L. *LoRa Alliance*. 2018. <https://lora-alliance.org/>.

ALLIANCE, T. Z. *Low-power, low-cost, low-complexity networking for the Internet of Things*. 2002. <https://www.zigbee.org/zigbee-for-developers/network-specifications/>.

ASHTON, K. That 'internet of things' thing. *RFiD Journal*, v. 22(7), p. 97–114, 2009.

ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: A survey. *Computer Networks*, v. 54, n. 15, p. 2787–2805, 2010. ISSN 13891286.

AYA, T. et al. Análise dos Impactos Ambientais na produção de Energia dentro do Planejamento Integrado De Recursos. *Departamento de Engenharia de Construção Civil- Escola Politécnica- Universidade de São Paulo*, v. 1, n. 1, p. 14, 2012.

BARAI, G. R.; KRISHNAN, S.; VENKATESH, B. Smart metering and functionalities of smart meters in smart grid - a review. In: . London, ON, Canada: IEEE, 2015.

BERTOLDI, B. *DPAE desenvolve mapa do consumo de energia elétrica na UFSC*. 2014. <https://noticias.ufsc.br/2014/07/dpae-desenvolve-mapa-do-consumo-de-energia-eletrica-na-ufsc/>.

BEST, J. *Getting the Wemos D1 Mini board to work on Mac OS 10.13 (High Sierra)*. 2018. <https://medium.com/@jimgbest/getting-the-wemos-d1-mini-board-to-work-on-mac-os-10-13-high-sierra-f30324d82db2>.

BRIGHT, P. *Visual Studio now supports debugging Linux apps; Code editor now open source*. 2015. <https://arstechnica.com/information-technology/2015/11/visual-studio-now-supports-debugging-linux-apps-code-editor-now-open-source/>.

CLAUSEN, T.; JACQUET, P. Optimized Link State Routing Protocol (OLSR). p. 1–75, 2003. ISSN 2070-1721. <https://www.rfc-editor.org/info/rfc3626>.

DAVIES, G. et al. ITU-T Y.2060 TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU. *Inorganic Chemistry*, v. 25, n. 25, p. 4479–4487, 2012. ISSN 1520510X.

DELICATO, T. B. P. F. P. F. C. Plataformas para a Internet das Coisas. 2015.

DEMETRAS, E. *SCT-013 – Sensor de Corrente Alternada com Arduino*. 2017. <https://portal.vidadesilicio.com.br/sct-013-sensor-de-corrente-alternada/>.

DUVANDER, A. *JSON's Eight Year Convergence With XML*. 2013. <https://www.programmableweb.com/news/jsons-eight-year-convergence-xml/2013/12/26>.

ESPRESSIF. *ESP-MESH*. 2017. <https://docs.espressif.com/projects/esp-idf/en/latest/api-guides/mesh.html>.

ESPRESSIF. *ESP Mesh*. 2018. <https://www.espressif.com/en/products/software/esp-mesh/overview>.

EUROTECH; IBM. *MQTT specification version 3.1*. 2010. <public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html>.

FAN, T.; CHEN, Y. A Scheme of Data Management in the Internet of Things. In: *2nd IEEE International Conference on Network Infrastructure and Digital Content*. [S.l.: s.n.], 2010.

FARIAS, M. M. et al. Análise De Queimadas Na Região Amazônica Através De Redes Sensoriais. 2005.

FERNANDES, N. C. et al. Ataques e Mecanismos de Segurança em Redes Ad Hoc. 2006. <http://professor.ufabc.edu.br/ joao.kleinschmidt/aulas/seg2011/adhoc.pdf>.

FIELDING, R. T. Architectural Styles and the Design of Network-based Software Architectures. *Journal of Visual Languages & Computing*, v. 11, n. 3, p. 287–301, 2000.

FIGUEIREDO, F. L. et al. Análise de desempenho de protocolos de roteamento para redes Ad Hoc sem fio. 2006.

FLEESON, W. et al. MONITORAMENTO E REVISÃO DO PLANO DE LOGÍSTICA SUSTENTÁVEL DA UFSC. *Journal of Personality and Social Psychology*, v. 1, n. 1, p. 1188–1197, 2017.

GARTNER, I. Hype Cycle for Emerging Technologies Identifies the Computing Innovations That Organizations Should Monitor. 2015.

HUDSON, G. L. T. *OPEN ENERGY MONITOR*. 2016. <https://learn.openenergymonitor.org/electricity-monitoring/ct-sensors/interface-with-arduino>.

IETF. *Constrained RESTful environments: charter core*. 2010. <https://datatracker.ietf.org/doc/charter-ietf-core/>.

IETF. *Request for Comments: 7252*. 2014. 1 p. <https://tools.ietf.org/html/rfc7252>.

JAFFEY, T. *MQTT and CoAP - IoT Protocols*. 2014. <http://www.eclipse.org/community/eclipse-newsletter/2014/february/article2.php>.

JOSÉ, O.; JR, L.; PRZYBYSZAUTOR, A. L. Infra-Estrutura E Roteamento Em Redes Wireless Mesh. v. 02, p. 3–5, 2007.

KUROSE, J. F.; ROSS, K. W. *Computer Networking: A Top-Down Approach*. 6th editio. ed. [S.l.]: Pearson, 2012.

LEA TRYSTAN. HUDSON, G. *OPEN ENERGY MONITOR*. 2016. <https://learn.openenergymonitor.org/electricity-monitoring/ct-sensors/interface-with-arduino>.

LISBOA, R.; ROVERE, D. Protótipo de um Sistema Inteligente de Monitoramento do Consumo de Energia Elétrica de uma Residência. 2016.

LOUREIRO, A. A. et al. Redes de Sensores Sem Fio. 2003.

MARTINS, I. R.; ZEM, J. L. Estudo dos Protocolos de Comunicação MQTT e CoAP para Aplicações Machine-to-Machine e Internet das Coisas. *Revista Tecnológica da Fatec Americana*, v. 3, n. 1, p. 24, 2016. ISSN 2446-7049.

Maxim Integrated Products. *DS18B20*. 2018. 1–21 p.

MESSIAS, A. A. Redes Inteligentes de Energia - Smart Grids. *Encontro Nacional do Colégio de Engenharia Electrotécnica*, 2009.

MIORANDI, D. et al. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, Elsevier, v. 10, p. 1497–1516, 2012.

MQTT.org. *MQTT*. 1999. <http://mqtt.org/faq>.

MURTA, G. *Conhecendo o ESP32 – Introdução (1)*. 2018. <https://jgamblog.wordpress.com/2018/02/05/conhecendo-o-esp32-introducao-01/>.

NODERED.ORG. *Node-RED - Flow-based programming for the Internet of Things*. 2013. <https://nodered.org/>.

OLSON, N. The Internet of things. *New Media & Society*, v. 18, p. 680–682, 2016. <http://dx.doi.org/10.1177/1461444815621893a>.

PERES, B. S. et al. Internet das Coisas: da Teoria à Prática.

PERKINS, C.; BELDING-ROYER, E.; DAS, S. *Ad Hoc On-Demand Distance Vector (AODV) routing*. 2003. <http://www.ietf.org/rfc/>.

PETERSON, L. L.; DAVIE, B. S. *Computer Networks, Fifth Edition: A Systems Approach*. 5th editio. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

RAMANATHAN, R.; REDI, J. A brief overview of ad hoc networks: challenges and directions. *IEEE Communications Magazine*, v. 40, n. 5, p. 20–22, 2002. ISSN 0163-6804. <http://ieeexplore.ieee.org/document/1006968/>.

SANTOS, R. *ESP8266 DS18B20 Temperature Sensor Web Server with Arduino IDE*. 2016. <https://randomnerdtutorials.com/esp8266-ds18b20-temperature-sensor-web-server-with-arduino-ide/>.

SHELBY, Z. *CoAP: the Web of things protocol*. 2014. <https://pt.slideshare.net/zdshelby/coap-tutorial>.

SILVA, Z. S. Construindo roteadores Wi-Mesh com GNU/Linux E OLSR. 2015.

SYSTEMS, E. ESP32 Series. 2018. ISSN 00070610.

T., S. N. K. K. S. D.; MUKHOPADHYAY, S. C. Towards the Implementation of IoT for Environmental Condition Monitoring in Homes. *IEEE SENSORS JOURNAL*, v. 13, n. 10, 2013. ISSN 1530437X.

TANENBAUM, A. *Computer Networks*. 4th editio. ed. [S.l.]: Prentice Hall Professional Technical Reference, 2002.

TANENBAUM, A. S. *Computer Networks*. [s.n.], 1996. 349–351 p. ISSN 13891286. ISBN 0130661023. <http://www.ietf.org/rfc/rfc169.txt>.

TCU. Tribunal de contas da união tc 022.649/2009-4. p. 1–48, 2013.

TEIXEIRA, E. R. D. *Wireless Mesh Networks*. 2004.

UFSC. *Eficiência Energética*. 2013. <http://ufscsustentavel.ufsc.br/eficiencia-energetica/>.

UFSC. *O Programa*. 2013. <http://ufscsustentavel.ufsc.br/o-programa/>.

UFSC, D. *Localização Faturas UFSC*. 2014. <https://noticias.ufsc.br/2014/07/dpae-desenvolve-mapa-do-consumo-de-energia-eletrica-na-ufsc/>.

UFSC, D. *Quanto a UFSC gasta com energia elétrica?* 2017. <http://dpae.seoma.ufsc.br/2017/02/01/gasto-ufsc-energia-eletrica/>.

YAN, L. et al. *THE INTERNET OF THINGS: From RFID to the Next-Generation Pervasive Networked Systems*. 1st editio. ed. Broken Sound Parkway NW: Auerbach Publications, 2008. 336 p.

YANG, C.-c.; TSENG, L.-p. Fisheye Zone Routing Protocol for Mobile Ad Hoc Networks. v. 30, p. 1–6, 2004.

YHDC. Split core current transformer. *Split core current transformer*, p. 7929499, 2010.

YHDC. *Current Transformer*. 2018. <http://en.yhdc.com/nav/Product-12.html>.

ZHENG, J.; GAO, D. W.; LIN, L. Smart Meters in Smart Grid: An Overview. In: . Denver, CO, USA: IEEE, 2013.

APÊNDICE A – Códigos fonte

A.1 APÊNDICE A

Código fonte dos nodos

```
#include "painlessMesh.h"
#include "EmonLib.h"
#include "OneWire.h"
#include "DallasTemperature.h"

#define ONE_WIRE_BUS 15

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

EnergyMonitor emon1;

#define MESH_SSID "MESH_NET_NODE"
#define MESH_PASSWORD "somethingSneaky"
#define MESH_PORT 5555

void sendMessage();
void calculateEnergy();
void receivedCallback(uint32_t from, String &msg);
void newConnectionCallback(uint32_t nodeId);
void changedConnectionCallback();
void nodeTimeAdjustedCallback(int32_t offset);
void delayReceivedCallback(uint32_t from, int32_t delay);

painlessMesh mesh;
bool calc_delay = false;
SimpleList<uint32_t> nodes;

uint32_t dest = 3816823837;

Task taskSendMessage(TASK_SECOND * 5, TASK_FOREVER, &
    sendMessage);

bool onFlag = false;

hw_timer_t *timer = NULL;

void IRAM_ATTR resetModule() {
  ets_printf("(watchdog) reiniciar\\n");
  esp_restart_noos();
}

void setup()
{
  Serial.begin(115200);
  sensors.begin();

```

```
47    analogReadResolution(ADC_BITS);
48    emon1.voltage(39, 230, 1.7);
49    emon1.current(36, 91);
50
51    mesh.init(MESH_SSID, MESH_PASSWORD, MESH_PORT);
52    mesh.onReceive(&receivedCallback);
53    mesh.onNewConnection(&newConnectionCallback);
54    mesh.onChangedConnections(&changedConnectionCallback);
55    mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
56    mesh.onNodeDelayReceived(&delayReceivedCallback);
57
58    mesh.scheduler.addTask(taskSendMessage);
59    taskSendMessage.enable();
60
61    timer = timerBegin(0, 80, true);
62    timerAttachInterrupt(timer, &resetModule, true);
63    timerAlarmWrite(timer, 7000000, true);
64    timerAlarmEnable(timer);
65  }
66
67  void loop()
68  {
69    mesh.update();
70    timerWrite(timer, 0);
71  }
72
73  void sendMessage()
74  {
75    sensors.requestTemperatures();
76    float temp = sensors.getTempCByIndex(0);
77
78    emon1.calcVI(2000, 2000);
79    float apparentPower = emon1.apparentPower;
80    float supplyVoltage = emon1.Vrms;
81    float Irms = emon1.Irms;
82    float energyHour = apparentPower*5;
83
84    String payload = "{\"d\":{";
85    payload += "\"PotA\":";
86    payload += apparentPower;
87
88    payload += ",\"VolA\":";
89    payload += supplyVoltage;
90
91    payload += ",\"CurA\":";
92    payload += Irms;
93
94    payload += ",\"EneA\":";
95    payload += energyHour;
96
97    payload += ",\"TemA\":";
98    payload += temp;
```

```
99    payload += "}}";
100
101   Serial.println(payload);
102   mesh.sendSingle(dest, payload);
103 }
104
105 void receivedCallback(uint32_t from, String &msg)
106 {
107   Serial.printf("Received: %s from: %u\n", msg.c_str(), from
        );
108 }
109
110 void newConnectionCallback(uint32_t nodeId)
111 {
112   onFlag = false;
113   Serial.printf("--> startHere: New Connection, nodeId = %u\
        n", nodeId);
114 }
115
116 void changedConnectionCallback()
117 {
118   Serial.printf("Changed connections %s\n", mesh.
        subConnectionJson().c_str());
119   onFlag = false;
120
121   nodes = mesh.getNodeList();
122
123   Serial.printf("Num nodes: %d\n", nodes.size());
124   Serial.printf("Connection list:");
125
126   SimpleList<uint32_t>::iterator node = nodes.begin();
127   while (node != nodes.end())
128   {
129     Serial.printf(" %u", *node);
130     node++;
131   }
132   Serial.println();
133   calc_delay = true;
134 }
135
136 void nodeTimeAdjustedCallback(int32_t offset)
137 {
138   Serial.printf("Adjusted time %u. Offset = %d\n", mesh.
        getNodeTime(), offset);
139 }
140
141 void delayReceivedCallback(uint32_t from, int32_t delay)
142 {
143   Serial.printf("Delay to node %u is %d us\n", from, delay);
144 }
```

Código fonte do Gateway

```
#include <painlessMesh.h>

#define   MESH_SSID         "MESH_NET_NODE"
#define   MESH_PASSWORD     "somethingSneaky"
#define   MESH_PORT         5555

void receivedCallback(uint32_t from, String & msg);
void newConnectionCallback(uint32_t nodeId);
void nodeTimeAdjustedCallback(int32_t offset);
void delayReceivedCallback(uint32_t from, int32_t delay);

painlessMesh  mesh;
bool calc_delay = false;
SimpleList<uint32_t> nodes;

bool onFlag = false;

hw_timer_t *timer = NULL;

void IRAM_ATTR resetModule() {
  ets_printf("(watchdog) reiniciar\n");
  esp_restart_noos();
}

void setup() {

  Serial.begin(115200);
  mesh.init(MESH_SSID, MESH_PASSWORD, MESH_PORT);
  mesh.onReceive(&receivedCallback);
  mesh.onNewConnection(&newConnectionCallback);
  mesh.onChangedConnections(&changedConnectionCallback);
  mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
  mesh.onNodeDelayReceived(&delayReceivedCallback);

  timer = timerBegin(0, 80, true);
  timerAttachInterrupt(timer, &resetModule, true);
  timerAlarmWrite(timer, 7000000, true);
  timerAlarmEnable(timer);
}

void loop() {
  mesh.update();
  timerWrite(timer, 0);
}

void receivedCallback(uint32_t from, String & msg) {
  Serial.printf("Received: %s from: %u\n", msg.c_str(), from
    );
  Serial.println(msg.c_str());
  delay(10);
```

```
50 }
51
52 void newConnectionCallback(uint32_t nodeId) {
53   onFlag = false;
54    Serial.printf("--> startHere: New Connection, nodeId = %u
       \n", nodeId);
55 }
56
57 void changedConnectionCallback() {
58   onFlag = false;
59   nodes = mesh.getNodeList();
60   void changedConnectionCallback();
61
62   SimpleList<uint32_t>::iterator node = nodes.begin();
63   while (node != nodes.end()) {
64     node++;
65   }
66   calc_delay = true;
67 }
68
69 void nodeTimeAdjustedCallback(int32_t offset) {
70   Serial.printf("Adjusted time %u. Offset = %d\n", mesh.
       getNodeTime(), offset);
71 }
72
73 void delayReceivedCallback(uint32_t from, int32_t delay) {
74   Serial.printf("Delay to node %u is %d us\n", from, delay);
75 }
```

Código fonte do Bridge

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>

const char* ssid = "XXXXX";
const char* password = "XXXXX";

#define ORG "is3q17"
#define DEVICE_TYPE "ESP32"
#define DEVICE_ID "40A5EFD6E798"
#define TOKEN "JZTa4aIl9L_E3axaPs"

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

const char publishTopic[] = "iot-2/evt/status/fmt/json";
const char responseTopic[] = "iotdm-1/response";
const char manageTopic[] = "iotdevice-1/mgmt/manage";
const char updateTopic[] = "iotdm-1/device/update";
const char rebootTopic[] = "iotdm-1/mgmt/initiate/device/reboot";

void callback(char* topic, byte* payload, unsigned int payloadLength);

WiFiClient wifiClient;
PubSubClient client(server, 1883, callback, wifiClient);

int publishInterval = 1000;
long lastPublishMillis;

void wifiConnect();
void mqttConnect();
void initManagedDevice();

void setup() {
  Serial.begin(115200);
  while (!Serial);
  Serial.println("Serial ok");

  wifiConnect();
  mqttConnect();
  initManagedDevice();
}

void loop() {
  Serial.setTimeout(2);
```

```
48   if (Serial.available() > 0)
49   {
50     String payload = Serial.readString();
51
52     if (strstr(payload.c_str(), "Pot") != NULL)
53     {
54       Serial.print("Sending payload: "); Serial.println(
     payload);
55
56       if (client.publish(publishTopic, (char*) payload.c_str
     ())) {
57         Serial.println("Publish OK");
58       } else {
59         Serial.println("Publish FAILED");
60       }
61     }
62   }
63   if (!client.loop()) {
64     mqttConnect();
65     initManagedDevice();
66   }
67 }
68
69 void wifiConnect() {
70   Serial.print("Connecting to "); Serial.print(ssid);
71   WiFi.begin(ssid, password);
72   while (WiFi.status() != WL_CONNECTED) {
73     delay(500);
74     Serial.print(".");
75   }
76   Serial.print("nWiFi connected, IP address: "); Serial.
     println(WiFi.localIP());
77 }
78
79 void mqttConnect() {
80   if (!!!client.connected()) {
81     Serial.print("Reconnecting MQTT client to "); Serial.
     println(server);
82     while (!!!client.connect(clientId, authMethod, token)) {
83       Serial.print(".");
84       delay(500);
85     }
86     Serial.println();
87   }
88 }
89
90 void initManagedDevice() {
91   if (client.subscribe("iotdm-1/response")) {
92     Serial.println("subscribe to responses OK");
93   } else {
94     Serial.println("subscribe to responses FAILED");
95   }
```

```
96
97    if (client.subscribe(rebootTopic)) {
98      Serial.println("subscribe to reboot OK");
99    } else {
100     Serial.println("subscribe to reboot FAILED");
101   }
102
103   if (client.subscribe("iotdm-1/device/update")) {
104     Serial.println("subscribe to update OK");
105   } else {
106     Serial.println("subscribe to update FAILED");
107   }
108
109   StaticJsonBuffer<300> jsonBuffer;
110   JsonObject& root = jsonBuffer.createObject();
111   JsonObject& d = root.createNestedObject("d");
112   JsonObject& metadata = d.createNestedObject("metadata");
113   metadata["publishInterval"] = publishInterval;
114   JsonObject& supports = d.createNestedObject("supports");
115   supports["deviceActions"] = true;
116
117   char buff[300];
118   root.printTo(buff, sizeof(buff));
119   Serial.println("publishing device metadata:"); Serial.
        println(buff);
120   if (client.publish(manageTopic, buff)) {
121     Serial.println("device Publish ok");
122   } else {
123     Serial.print("device Publish failed:");
124   }
125 }
126
127 void callback(char* topic, byte * payload, unsigned int
        payloadLength) {
128   Serial.print("callback invoked for topic: "); Serial.
        println(topic);
129
130   if (strcmp (responseTopic, topic) == 0) {
131     return;
132   }
133
134   if (strcmp (rebootTopic, topic) == 0) {
135     Serial.println("Rebooting...");
136     ESP.restart();
137   }
138
139   if (strcmp (updateTopic, topic) == 0) {
140     handleUpdate(payload);
141   }
142 }
143
144 void handleUpdate(byte * payload) {
```

```
145   StaticJsonBuffer<300> jsonBuffer;
146   JsonObject& root = jsonBuffer.parseObject((char*)payload);
147   if (!root.success()) {
148     Serial.println("handleUpdate: payload parse FAILED");
149     return;
150   }
151   Serial.println("handleUpdate payload:"); root.
        prettyPrintTo(Serial); Serial.println();
152
153   JsonObject& d = root["d"];
154   JsonArray& fields = d["fields"];
155   for (JsonArray::iterator it = fields.begin(); it != fields
        .end(); ++it) {
156     JsonObject& field = *it;
157     const char* fieldName = field["field"];
158     if (strcmp (fieldName, "metadata") == 0) {
159       JsonObject& fieldValue = field["value"];
160       if (fieldValue.containsKey("publishInterval")) {
161         publishInterval = fieldValue["publishInterval"];
162         Serial.print("publishInterval:"); Serial.println(
      publishInterval);
163       }
164     }
165   }
166 }
```