Matheus Valente da Silva

IMPLEMENTAÇÃO DE UMA REDE DE SENSORES 6LOWPAN

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Engenharia Eletrônica.
Orientador: Prof. Dr. Richard Demo Souza

Florianópolis
2018

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

da Silva, Matheus Valente
Implementação de uma rede sensores 6LoWPAN / Matheus Valente da Silva ; orientador, Richard Demo Souza, 2018.
75 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Eletrônica, Florianópolis, 2018.

Inclui referências.

1. Engenharia Eletrônica. 2. Rede Sensores. 3. 6LOWPAN. I. Demo Souza, Richard. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Eletrônica. III. Título.

Matheus Valente da Silva

IMPLEMENTAÇÃO DE UMA REDE DE SENSORES 6LOWPAN

Este Trabalho foi julgado adequado para obtenção do Título de Bacharel em Engenharia Eletrônica e aprovado em sua forma final pela Banca Examinadora

Florianópolis, 3 de dezembro de 2018.

Prof. Jeferson Luiz Brum Marques, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Richard Demo Souza, Dr.
Orientador
Universidade Federal de Santa Catarina

Prof. Arliones Hoeller Jr,
Instituto Federal de Santa Catarina

Prof. Mario de Noronha Neto, Dr.
Instituto Federal de Santa Catarina

Este trabalho é dedicado aos meus queridos pais.

AGRADECIMENTOS

À Universidade Federal de Santa Catarina que me proporcionou inúmeras oportunidades de aprendizagem ao longo dessa graduação.

À CAPES que fomentou o edital 180 do Ciência sem Fronteiras me permitindo estudar na The University of Akron e realizar o treinamento acadêmico no Illinois Institute of Technology onde comecei meu estudo de comunicações.

Ao professor Richard Demo Souza pela orientação e a oportunidade de realizar esse trabalho.

Ao professor Guilherme Moritz pela ajuda e disponibilidade.

À minha namorada, Maria Julia, pelo apoio e incentivo que muito me ajudou durante o curso.

Aos amigos Allan, Daniela, Handerson, Jaclyn, Raul, Thiago, Victor e Vinicius e tantos outros que muito me ajudaram nessa jornada.

Por fim gostaria de agradecer ao meus pais pela educação, apoio, incentivo e amor.

RESUMO

Este trabalho apresenta os requisitos de uma rede de sensores sem fio, bem como uma revisão dos protocolos necessários para a implementação de uma rede de sensores 6LoWPAN. Apresenta também um comparativo entre as duas mais importantes categorias de redes para a internet das coisas. Por fim, é demonstrada a implementação de uma rede baseada na tecnologia 6LoWPAN.

Palavras-chave: Rede de sensores sem fios. 6LoWPAN. Internet das coisas.

ABSTRACT

This work presents the requirements of a wireless sensor network, as well as an overview of the needed protocols to implement such a network using the 6LoWPAN protocol stack. A comparison of the two most important categories of networks for the Internet of Things is also presented. To conclude it is demonstrated the implementation of a 6LOWPAN network.

Keywords: Wireless sensor network. 6LoWPAN. Internet of Things.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

6LOWPAN – IPv6 over Low-power Wireless Personal Area Networks
BPSK – Binary Phase-Shift Keying
CCS – Code Composer Studio
DNS – Domain Name System
DODAG – Destination Oriented Directed Acyclic Graph
DSSS – Direct Sequence Spread Spectrum
FFD – Full Function Device
HTTP – Hypertext Transfer Protocol
IDE – Integrated Development Environment
IEEE – Institute of Electrical and Electronics Engineers
IETF – Internet Engineering Task Force
IP – Internet Protocol
IPv4 – Internet Protocol version 4
IPv6 – Internet Protocol version 6
ISM – Industrial Sientific and Medical
ISO – International Organization for Standardization
LLN – Low power and Lossy Network
LPWANs – Low power wide area network
MAC – Medium Access Control
MQTT – Message Queuing Telemetry Transport
MQTT-SN – Message Queuing Telemetry Transport Sensor Network
MTU – Maximum Transmission Unit
OSI – Open System Interconnection
OQPSK – Offset quadrature phase-shift keying
PSSS – Parallel Sequence Spread Spectrum
RFD – Reduced Function Device
RPL – Routing Protocol over Low-Power and Lossy Networks
RSMB – Really Small Message Broker
SLAAC – Stateless Address Autoconfiguration
TCP – Transmission Control Protocol
TSCH – Time Synchronized Channel Hopping,

UDP – User Datagram Protocol
WG – Working Groups
WPAN – Wireless Personal Area Network
WSN – Wireless Sensor Network

SUMÁRIO

1 INTRODUÇÃO

Previsões relacionadas com o cenário de Internet das Coisas ou IoT (do inglês *Internet of Things*) esperam mais de 28 bilhões de dispositivos conectados até 2021[1]. Além disso, de acordo com [2] IoT deve gerar 344 bilhões de dólares em receita em 2019. A cada ano mais dispositivos estão conectados à Internet com a habilidade de medir e controlar o ambiente ao seu redor. Com isso uma enorme gama de aplicações já está no mercado.

Por exemplo, o projeto iMETland está testando e validando uma solução com redes de sensores [3] para o tratamento inteligente de água em pequenas cidades e regiões isoladas em quatro países (Espanha, Dinamarca, Argentina e México) [4]. O projeto tem como principais objetivos fazer a leitura de diversos parâmetros a cada 15 minutos, mandando os dados para a nuvem onde são extraídas informações para inferir padrões e gerar alertas e previsões. As redes de sensores desse projeto são implementadas em regiões isoladas sem acesso a fontes de alimentação. Para realizar a comunicação com o roteador esta rede utiliza o padrão IEEE802.15.4 [5], o qual vai ser tratado ao longo deste trabalho e utilizado na implementação. O roteador por sua vez utiliza o 3G para se conectar à nuvem que está ligada à uma aplicação para a análise de dados e o controle do tratamento.

Outro projeto, o fPerception, foi desenvolvido na Polônia para monitorar e aprimorar a atividade portuária e reduzir o impacto ambiental do porto de Gdansk [6]. Utilizando uma série de sensores da Libelium [7], o projeto se propôs a medir a temperatura, umidade, pressão, concentração de amônia, concentração de sulfato de hidrogênio, concentração de fosfina, velocidade e direção do vento e a precipitação de chuva. Todas essas medidas são enviadas através de uma rede 4G para um roteador que utiliza o protocolo MQTT [8], que também utilizaremos ao longo deste trabalho, para publicar os dados na

nuvem. Os dados por sua vez são processados e apresentados por meio de um aplicativo, através do qual com o objetivo de poupar tempo dos interessados, aumentar a segurança dos trabalhadores com um sistema de aviso de poluição e poupar dinheiro através das regulações europeias de incentivo à redução de poluição.

1.1 OBJETIVOS

1.1.1 Objetivo geral

As aplicações acima demonstram a importância e capacidade impressionante das redes sensores de impactar a sociedade. Com isso em mente esse trabalho tem o objetivo de revisar alguns dos protocolos desenvolvidos e utilizados para a implementação dessas redes, assim como colocar em funcionamento uma rede 6LoWPAN [9].

1.1.2 Objetivos específicos

Os objetivos específicos deste trabalho podem ser divididos nos seguintes itens:

1) Apresentar os requisitos de uma rede de sensores sem fio.
2) Apresentar uma revisão dos protocolos a serem utilizados.
3) Implementar uma rede de sensores a partir da qual pode-se criar diversas aplicações.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 REDE DE SENSORES SEM FIO

Uma rede de sensores sem fio, ou WSN (do inglês *Wireless Sensor Network*) é definida como uma rede auto-organizável de sensores que se comunicam utilizando de rádio para obter informações do mundo físico [10].

Em geral cada nó sensor consiste de um processador e memória limitados, um rádio, fonte de energia e sensores. Esses elementos cumprem as seguintes funções [10]:

a) Processador: responsável por processar as informações obtidas localmente e/ou as informações dos nós vizinhos. Suas limitações computacionais se dão devido à exigência de baixo consumo de energia e pequeno tamanho.
b) Memória: é utilizada para armazenar as informações coletadas e o programa a rodar no processador. Também é limitada pelo tamanho e custo.
c) Rádio: os rádios utilizados tipicamente têm baixa taxa de transmissão, entre 10 e 250 kbps, e curto alcance, geralmente menos de 100 metros. Como a comunicação é a tarefa que mais consome energia é necessário utilizar técnicas e protocolos eficientes para aumentar a autonomia do nó.
d) Fonte de energia: os nós sensores devem ter uma longa vida útil. Por isso, podem ser projetados com baterias recarregáveis e painéis solares para que eles possam agir de modo autônomo o maior período de tempo possível.
e) Sensores: uma rede de sensores pode utilizar uma enorme variedade de sensores de modo a ter uma melhor leitura do ambiente desejado. Podem ser pequenos, baratos e de fácil aquisição ou sensores

mais complexos, de alto custo e de difícil implementação como os capazes de medir quantidades de metais pesados.

O objetivo de uma WSN é espalhar os nós por uma região para coletar informações para observar, entender e analisar uma experiência ou um fenômeno rapidamente. Podemos ter um alto número de sensores na região de interesse. Assim, inúmeras aplicações podem ser desenvolvidas, sejam elas para monitorar o meio ambiente, a saúde, segurança, e etc, criando assim casas, prédios e ambientes altamente inteligentes.

De acordo com [10], as seguintes características técnicas tornam as WSN uma tecnologia atrativa:

a) Redes sem fio: Os nós se comunicam entre si através de rádio para difundir e processar os dados coletados pelos sensores. Dependendo da configuração da rede alguns nós podem ser utilizados como relay para repassar os dados a outro nó. Nesse caso a rede é chamada de multi-hop. Se os nós se comunicam apenas com o diretamente com o gateway a rede é chamada de single-hop. As redes sem fio são capazes de monitorar locais remotos e de difícil acesso, ou ambientes onde dispositivos com fios podem causar uma perturbação indesejada. Além disso, é estimado que as redes sem fio podem reduzir o custo de instalação em até 80% em relação às redes com fio [10].
b) Auto-organização: os nós se organizam em uma rede ad-hoc, ou seja, eles não precisam de uma infraestrutura implementada para poder funcionar. Cada nó possui um algoritmo capaz de descobrir os seus vizinhos e assim reconhecer quais os nós que ele deve ouvir e se comunicar através do rádio.

Com isso as redes são de fácil implantação, manutenção e expansão.

c) Baixo consumo de energia: Um dos principais requisitos e características das WSN é o baixo consumo. Visto que é necessário para que as aplicações sejam implementadas em locais remotos e mantenham uma longa vida útil. Assim, muitos dispositivos utilizam técnicas de captação de energia para prolongar a vida útil, além de utilizar processadores e rádios de baixo consumo. Também são comuns esquemas para que o processador fique em *sleep mod*e o maior tempo possível.

Uma WSN geralmente contém três tipos de nós [10]:

a) Nó sensor: utilizado para capturar medidas do ambiente ao seu redor e transmitir para um nó chamado de estação base.

b) Estação base: esses tipos de nó são encarregados de coletar os dados captados pelos nós sensores e transmitir para um gateway que geralmente é um computador alimentado por uma fonte estável capaz de processar, armazenar e transmitir a informação obtida.

c) Atuadores: são nós sensores que possuem a capacidade de controlar o ambiente ao seu redor conforme as informações lidas ou recebidas. Um atuador pode por exemplo acender uma lâmpada quando a luminosidade de um ambiente é baixa, controlar a temperatura de uma estufa ou liberar produtos para manter o pH de uma piscina entre outras ações.

2.2 IPv6

O modelo OSI (do inglês Open System Interconnection), publicado pela ISO em 1984 [11], foi proposto para caracterizar e padronizar a comunicação de ponta a ponta por diferentes dispositivos. Com isso em mente, o modelo especifica sete camadas. Cada uma dessas camadas é responsável por uma parte da comunicação, Figura 2.1. As responsabilidades das camadas são definidas como:

1) Camada física: responsável por todo o tipo de mídia física seja com ou sem fio. Encarregada dos sinais eletromagnéticos, da modulação, codificação e etc.
2) Enlace: é a camada encarregada de fazer a comunicação nó a nó, recebendo e enviando os quadros da camada de rede. Tem o seu próprio endereço (MAC – Medium Access Control) que depende da tecnologia utilizada.
3) Rede: a internet utiliza protocolo IP nessa camada, que é responsável por entregar corretamente os dados recebidos da camada de transporte para o seu destinatário.
4) Transporte: é a camada acima da rede que lida com a coordenação de transferência de dados entre *hosts*. Quanto será enviado, a qual taxa, para onde, e etc. Os protocolos mais comuns nessa camada são TCP e UDP.
5) Sessão: responsável por coordenar a comunicação entre diferentes aplicações.
6) Apresentação: nesta camada os dados são convertidos do formato de rede, bytes, para um formato inteligível para o usuário final. Utilizando ASCII para textos, GIF para imagens e etc.
7) Aplicações: como o nome indica é a camada onde ficam as aplicações que fazem uso da internet.

Como exemplo de protocolos desta camada temos HTTP, DNS e o MQTT.

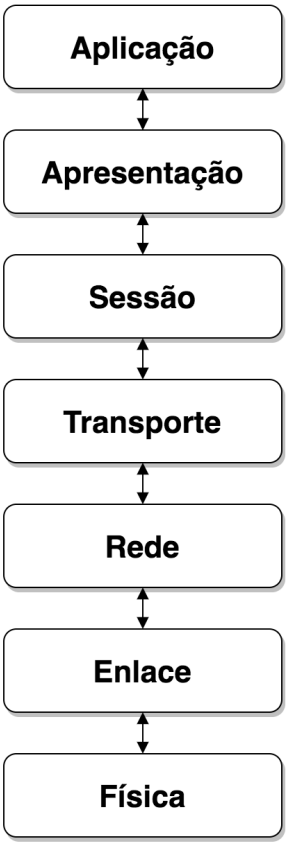

Figura 2.1 Modelo OSI

Com a crescente necessidade de conectar bilhões de dispositivos à internet o protocolo IP da camada de rede precisou ser atualizado para comportar tamanha demanda. Assim, foi criado o IPv6.

De acordo com [12] o IPv6 (do inglês *Internet Protocol version 6*) basicamente foi desenvolvido para aumentar a quantidade de endereços e a eficiência de roteamento da rede, visto que o IPv4 já não tinha mais endereços disponíveis. Para

isso foi utilizado um cabeçalho mais simples que o utilizado pela versão anterior com um esquema de endereços de 128 bits. Assim o IPv6 oferece até 2^{128} endereços.

Cada uma das camadas vistas acima adiciona alguma informação ao pacote IPv6 fazendo que ele sempre seja montado das camadas superiores para as inferiores como na Figura 2.2. Cada informação introduzida em determinada camada só pode ser processada pela mesma camada quando é realizado o processo de leitura do pacote.

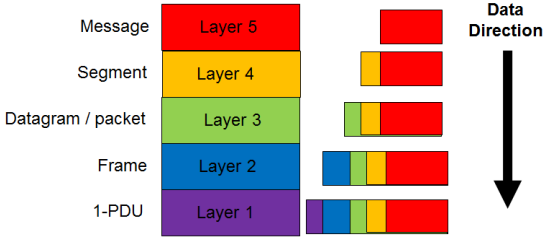

Figura 2.2 Fluxo de montagem do pacote IPv6 [10]

Ainda em [10] temos como o IPv6 foi aprimorado em relação ao IPv4. A quantidade de campos do cabeçalho foi reduzida de 12 para 8. O novo cabeçalho, Figura 2.3, tem um tamanho fixo de 40 bytes alinhado em 64 bits permitindo um encaminhamento de pacotes baseado em hardware mais rápido. Além do tamanho dos endereços que foi aumentado de 32 para 128.

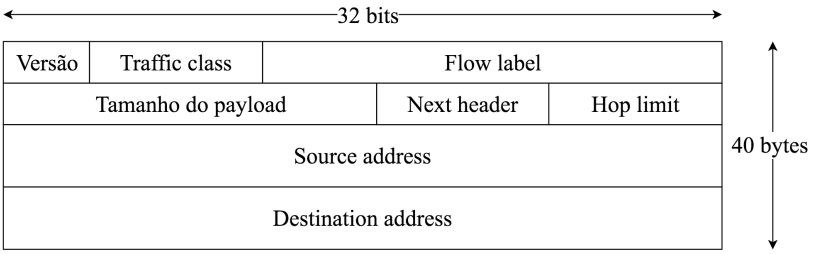

Figura 2.3 Cabeçalho básico IPv6

Os oito campos do cabeçalho acima são definidos da seguinte maneira:

1) Versão (4 bits): representa a versão do protocolo, 0110 no caso do IPv6.
2) Traffic class (8 bits): especifica o tipo de serviço de pacote para que o *router* saiba como tratar o mesmo.
3) Flow label (20 bits): usado para manter a ordem sequencial dos pacotes quando a mesma é importante.
4) Tamanho do *payload* (16 bits): define quanta informação o pacote terá no *payload*.
5) Next Header (8 bits): define se o pacote utilizará os cabeçalhos de extensão.
6) Hop limit (8 bits): define quantos saltos o pacote pode dar antes de ser descartado.
7) Endereço da fonte (128 bits): endereço do transmissor original do pacote.
8) Endereço de destino (128 bits): endereço do destino final do pacote.

Os endereços IPv6, bem como os IPv4, são divididos em quatro categorias:

1) Unicast: endereços utilizados para enviar pacotes a um único destinatário.
2) Multicast: endereços utilizados para enviar pacotes a vários destinatários.
3) Anycast: endereços utilizados para enviar pacotes ao destinatário mais próximo
4) Reserved: endereço ou grupo de endereços reservados para a documentação.

Os endereços IPv6 são descritos em 8 grupos de 16 bits escritos em hexadecimal, ou seja, 4 caracteres hexadecimal por grupo. Cada um desses grupos é separado por “:”. Além disso os zeros à esquerda de cada grupo podem ser omitidos e um ou mais grupos de zeros pode ser substituído por “::”, mas isso só pode ser feito uma única vez. Por exemplo:

1) 2001:0db8:85a3:0000:0000:8a2e:0370:7334
2) 2001:db8:85a3:0:0:8a2e:370:7334
3) 2001:db8:85a3::8a2e:370:7334

No primeiro caso temos um endereço IPv6, no segundo fizemos a omissão dos zeros à esquerda e no terceiro substituímos os grupos de zero. Caso houvesse dois grupos de zeros separados só poderíamos utilizar a notação “::” uma única vez. Exemplo:

1) 2801:84:0:0:fdc1:0:0:cd5f
2) 2801:84::fdc1:0:0:cd5f
3) 2801:84:0:0:fdc1::cd5f
4) 2801:84::fdc1::cd5f (notação errada)

Na lista acima em 1) temos a notação apenas com os zeros à esquerda omitidos. Em 2) e 3) temos as formas corretas de se comprimir os grupos de zeros. Não podemos utilizar a forma de 4) porque não saberíamos como estender o endereço.

Além disso também temos como definir um prefixo para as redes IPv6, onde definimos os primeiros 64 bits do endereço para indicar o prefixo da rede e os últimos 64 bits, chamados de Indentifier ID, para identificar os *hosts*. Com o endereço do último exemplo teríamos o seguinte prefixo de rede: 2801:84::/64, ou seja 2801:0084:0000:0000:0000:0000:0000:0000. Os prefixos de rede são sempre utilizados em LANs para definir os IPs dos dispositivos. Com isso eles têm um IP como na Figura 2.4.

Figura 2.4 Formato de IP em uma rede LAN com prefixo de rede

2.3 IEEE802.15.4

O padrão mais relevante para rádios de baixa potência é o IEEE802.15.4. Ele define o esquema de modulação utilizado na camada física, chamado de IEEE802.15.4 PHY, e na camada de controle de acesso do meio, chamada de IEEE802.15.4 MAC, a qual dita como o rádio se comportará dentro de uma rede, quando fala, com quem fala e qual canal utiliza para isso [13].

O protocolo IEEE802.15.4 PHY é um trade-off entre eficiência energética, alcance, taxa de dados [13]. Assim, ele é responsável pela transmissão e recepção de dados, ditando quais serão as bandas utilizadas e as técnicas de espalhamento e modulação.

Inicialmente, em 2003, o padrão definiu três bandas. A banda de 2.4 GHz ISM, a de 868 MHz para a Europa e a de 915 MHz para os Estados Unidos [13]. Ainda na primeira versão a banda de 2.4 GHz possuía 16 canais disponíveis com uma banda de guarda de 5 MHz entre eles, e utilizava a modulação O-QPSK com suporte a uma taxa de dados de 250 kbps. A banda europeia em 868 MHz tinha apenas um canal enquanto a americana em 915 MHz possuía dez com uma taxa de 20 kbps e 40 kbps, respectivamente. A modulação utilizada pelas bandas de 868/915 MHz era a BPSK.

Em 2006 o número de canais para a banda de 925 MHz foi ampliado para 30. Além de aumentar a taxa de dados de 868 e 915 MHz para 100 e 250 kbps, respectivamente, utilizando também a modulação O-QPSK baseada em DSSS [13]. No entanto, o padrão também definiu uma PHY opcional para 868/915 MHz onde era possível utilizar as modulações BPSK e

ASK baseadas em PSSS que tornava possível ambas alcançarem 250 kbps de taxa de transmissão de dados.

Na tabela 2.1 temos um quadro comparativo:

Tabela 2.1 Comparativo das frequências de banda do padrão 802.15.4

Frequência de Banda	**868 MHz**	**915 MHz**	**2.4 GHz**
Modulações disponíveis	ASK BPSK O-QPSK	ASK BPSK O-QPSK	O-QPSK
Número máximo de canais	1	30	16
Taxa máxima de dados	250 kbps	250 kbps	250kbps
Largura de banda	600 KHz	2 MHz	5 MHz

Fonte: [5]

Já o IEEE802.15.4 MAC incialmente foi projetado com o foco em redes do tipo estrela, onde todos os nós se comunicam diretamente com um coordenador. Porém quando usado em uma rede roteamento *multi-hop* levava os nós a ficarem com o rádio ligado 100% do tempo levando ao alto consumo de energia. Além disso o protocolo utilizava um único canal que causava instabilidade na rede devido à natureza do canal sem fio [13].

Em 2010 incorporou TSCH, Time Synchronized Channel Hopping, tornando-o altamente confiável e de baixa potência [8]. Com isso ele se torna mais adequado para as redes de sensores sem fio.

O trabalho em [9] define as principais características do 802.15.4 como:

1) Baixa taxa de transmissão: na frequência de 868/915 MHz são suportadas as taxas de 20 kbps até 250 kbps.
2) Baixo consumo de energia: pequenas baterias devem durar por vários meses.

3) Baixo custo: processamento embarcado limitado.
4) Curto alcance: o sinal pode ter um alcance de 10m a 100m.
5) Dispositivos com múltiplas funções: Full Function Devices (FFD) e Reduced Function Devices (RFD). Os FFD servem como coordenadores e roteadores e os RFD são os dispositivos finais da rede que passam a maior parte do tempo em *sleep mode*.
6) Dois modos de acesso ao canal: Em redes com *beacon* habilitado utiliza-se *slotted* CSMA/CA e em redes que não possuem o *beacon* habilitado utiliza-se *unslotted* CSMA/CA.
7) Suporta as topologias Mesh e Estrela como ilustradas na Figura 2.5.

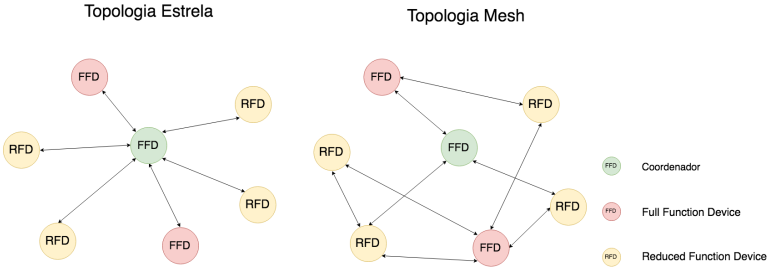

Figura 2.5 Topologias Estrela e Mesh

Segundo [10] a escolha do padrão 802.15.4 para a comunicação sem fio nas camadas físicas e MAC é muito promissora apesar de existirem outras boas opções como Low Power WiFi, Bluetooth Low Energy, DECT Ultra Low Energy, T G.9959 e o Near Field Communication.

2.4 6LoWPAN

Em 2007 a IEFT, Internet Engineering Task Force, criou o grupo IPv6 over Low power Wireless Personal Area Network (6LoWPAN) para definir um protocolo que transmitisse pacotes IPv6 através de rádios 802.15.4 [13]. Esse protocolo define uma camada de adaptação entre o IPv6 na Camada 3 e o padrão IEEE802.15.4, como exemplificado na Figura 2.6.

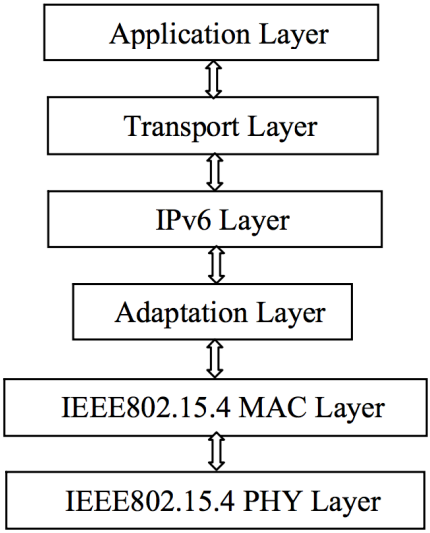

Figura 2.6 Referência de protocolos para o 6LoWPAN [10]

Com isso em mente, esta camada de adaptação foi projetada para alcançar os seguintes objetivos [10]:

1) Uma camada de fragmentação e remontagem: é desenvolvida para reduzir o tamanho do MTU, que é a maior quantidade dados que pode ser transmitida em um pacote, do IPv6 de 1280 bytes para 81 bytes que é o tamanho máximo suportado pelo padrão 802.15.4.
2) Compressão de cabeçalhos: é necessária visto que um cabeçalho IPv6 sem nenhuma extensão tem 40 bytes. Um pacote 802.15.4 tem 81 bytes. O que

deixaria apenas 41 bytes para os protocolos de transmissão como UDP e TCP. Sobrando ainda menos para transmitir informações de fato.

3) Autoconfiguração de endereços: especifica o método SLAAC (Stateless Address Autoconfiguration) para IPv6 reduzindo a necessidade de configuração nos nós [10].
4) Protocolo de roteamento *mesh*: É necessário o suporte a um roteamento *multi-hop*. Para isso foi desenvolvido o RPL, Routing Protocol over Low-Power and Lossy Networks [10].

Para que que fosse possível transmitir pacotes IPv6 através das redes 802.15.4, o 6LoWPAN definiu os seguintes formatos de cabeçalho [13]:

1) No 6LoWPAN: identifica quando um pacote não está de acordo com o protocolo e deve ser descartado.
2) Dispatch: utilizado para indicar um cabeçalho IPv6 comprimido.
3) Mesh Addressing: permite que os pacotes 802.15.4 sejam encaminhados na camada de enlace tornando uma rede single-hop em *multi-hop*.
4) Fragmentation: são utilizados quando os pacotes IPv6 não cabem em um único pacote 802.15.4 e precisam ser enviados em pedaços.

A compressão de cabeçalhos IPv6 é realizada através de duas codificações chamadas de LOWPAN_IPHC e LOWPAN_NHC. Ambas fazem a codificação baseada em estados compartilhados. A primeira sendo encarregada da compressão do cabeçalho básico e a segunda dos cabeçalhos de extensão como o cabeçalho UDP, por exemplo. Essas

codificações permitem que ambas categorias sejam comprimidas para até 2 bytes dependendo do caso.

O roteamento é um problema complexo para o 6LoWPAN, dadas as restrições de bateria, processamento, armazenamento e canais de alta perda. As redes de sensores também sofrem mudanças constantes em sua topologia, uma vez que novos nós podem ser adicionados a todo momento e cada um desses nós pode entrar em *sleep mode*.

Para isso foi desenvolvido o protocolo de roteamento RPL. Esse protocolo é capaz de criar redes e distribuir as informações de roteamento para os nós rapidamente, adaptando a topologia da rede de maneira eficiente [13].

Em uma rede RPL os nós geralmente estão conectados através de caminhos *multi-hop* até um nó *root* que costuma ser responsável pelo armazenamento das informações e controle dos nós. Cada um desses dispositivos *root* possui um DODAG (Destination Oriented Directed Acyclic Graph) que calcula a melhor rota para cada um dos nós levando em consideração a qualidade do canal, os atributos e status do nó e o valor de uma função que busca otimizar alguma característica. Assim a topologia da rede é montada utilizando um sistema de rank como métrica, o rank é dado pela distância em saltos que determinado nó está do *root*.

Essas características permitem que o RPL seja capaz de montar redes de diferentes complexidades. Durante a Seção 3, onde implementaremos uma rede, vamos utilizar a topologia mais simples. Essa é formada por apenas um DODAG *root* e é geralmente utilizada em redes de sensores que monitoram uma pequena área. Mas o RPL é capaz de formar redes com vários DODAG *roots* independentes, dividindo a rede da melhor maneira para diferentes aplicações.

Para a formação da rede o protocolo RPL possui quatro tipos de mensagens de controle que são formatadas de acordo com o ICMPv6, do inglês Internet Control Message Protocol version 6, e seu objeto base tem 24 bytes [14]:

1) DIS (DAG Information Solicitation): pede informações sobre a DODAG aos nós participantes.
2) DIO (DAG Information Object): envia as informações que permitem um que um novo nó descubra a rede e as suas configurações.
3) DAO (DAG Adverstisement Object): envia um pedido ao seu nó "pai" para partipar da rede.
4) DAO-ACK (DAO Acknowledgment): envia uma resposta ao pedido anterior podendo permitir ou não a participação do nó na rede.

A mensagem DIO por compartilhar as informações da DODAG costuma usar diversos campos adicionais ao objeto base. Chegando a ter, por exemplo, 89 bytes. Mas por utilizarem os pacotes IPv6 passam pela mesma compressão [15].

As redes RPL são construídas utilizando as mensagens descritas acima. Periodicamente os nós enviam mensagens DIO *multicast* para informar possíveis novos nós sobre a existência de uma rede. Caso um nó inicializado não receba nenhuma mensagem DIO ele envia uma mensagem DIS *multicast* para procurar uma rede. Quando um nó recebe uma mensagem DIS ele envia uma resposta com uma mensagem DIO. O novo nó quando recebe a mensagem DIO analisa as informações da DODAG e e envia uma mensagem DAO ao seu nó "pai". O nó pai então lhe envia um DAO-ACK aceitando o novo nó na rede. A Figura 2.7 demonstra como as mensagens de controle são utilizadas para a criação de uma rede simples.

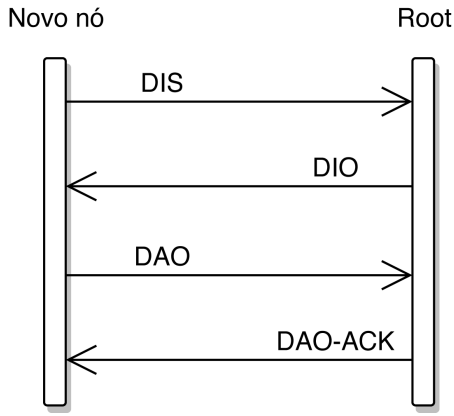

Figura 2.7 Exemplo de uma construção de uma rede com dois nós

Já em um caso com mais nós podemos ver a capacidade de otimização da rede em seis passos na Figura 2.8:

1) Root envia DIOs avisando os outros nós da existência da rede.
2) Os nós enviam um pedido DAO para participar da rede.
3) Root reconhece os nós como membros da rede respondendo com DAO-ACK e os mesmos calculam os seus ranks.
4) Os nós enviam DIOs informando sobre a rede e seu rank.
5) B e C percebem que podem melhorar seu rank escolhendo os nós A e C, respectivamente, como pais. Enviam pedidos DAO.
6) A e C aceitam os pedidos respondendo com uma mensagem DAO-ACK.

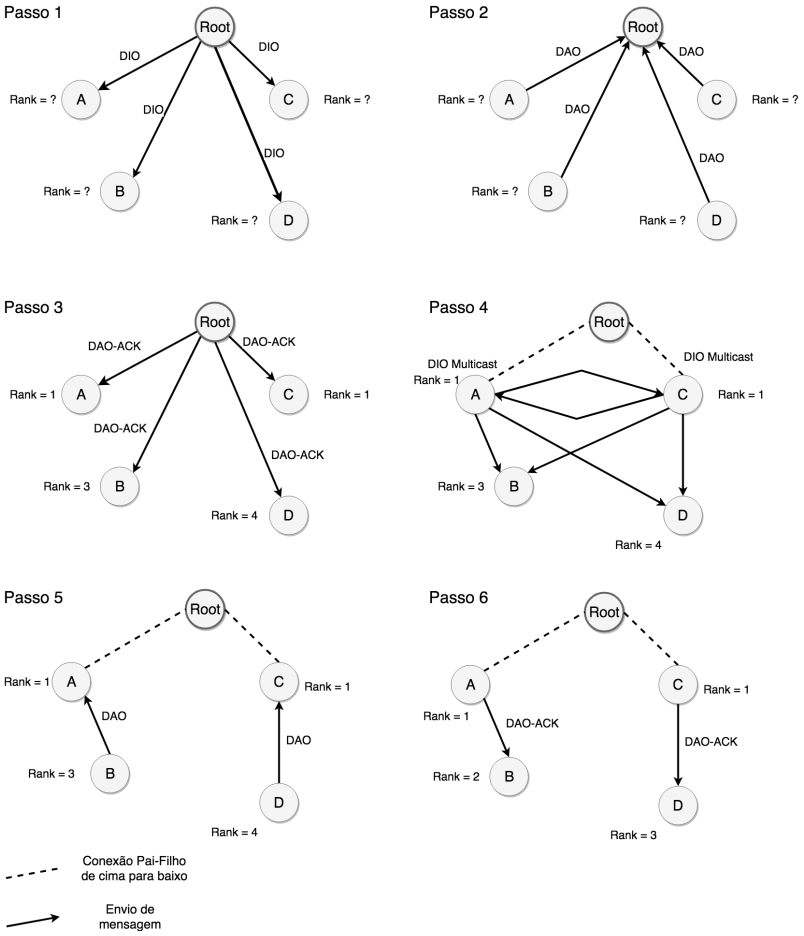

Figura 2.8 Otimização de uma rede RPL

2.5 COMPARAÇÃO ENTRE 6LOWPAN E LPWAN

Além da rede 6LoWPAN que foi abordada até agora, temos outros tipos de redes que são fundamentais no conceito de IoT, como as redes de longo alcance e baixa potência. As LPWANs (do inglês Low power wide area network) devem conter aproximadamente 25% dos dispositivos IoT no futuro,

sendo que já temos por exemplo SigFox e LoRaWAN (Long Range Wide Area Network) amplamente usadas ao redor do mundo [16]. De acordo com [17] SigFox já está presente em 45 países e LoRa já tem redes implementadas em 100 países [18] .

Segundo [11], as restrições das redes LPWAN são ainda mais exigentes que as 6LoWPAN em relação a custo, processamento, armazenamento e consumo de energia. Além disso, as LPWANs ainda não possuem suporte ao IPv6, no entanto a IETF criou um grupo em 2016 para adaptar o IPv6 a esta tecnologia.

Nas próximas subseções vamos abordar as diferenças entre as redes 6LoWPAN e LPWAN em relação à arquitetura de rede, camada física, camada de enlace e as camadas superiores.

2.5.1 Arquitetura de rede

A arquitetura de uma rede 6LoWPAN é formada por três elementos. Os *hosts*, que são os sensores e atuadores descritos anteriormente, os roteadores de fronteira ou *border routers* que fazem a conexão da rede 6LoWPAN com redes externas como a Internet, e os roteadores que enviam os pacotes de dados de *host* para o outro ou para o *border router*. A arquitetura de rede do 6LoWPAN pode ser vista na Figura 2.9.

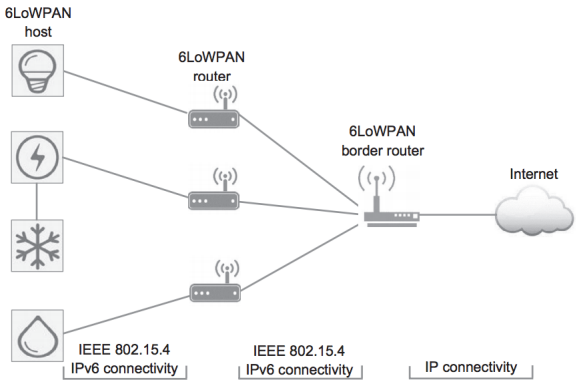

Figura 2.9 Arquitetura de rede 6LoWPAN [16]

Já a arquitetura de uma rede LPWAN, Figura 2.10, é uma rede tipo estrela onde os *host*s se conectam aos servidores através de uma comunicação *single-hop* e não são capazes de comunicar-se diretamente uns com os outros como em uma rede 6LoWPAN. A outra grande diferença entre as duas redes é a distância entre os pontos, essa sendo entre 10 e 100 metros para as redes 6LoWPAN e de até alguns quilômetros para as redes LPWAN. Com isso as redes SigFox e LoRa são capazes de cobrir cidades inteiras com alguns poucos gateways já as redes 6LoWPAN precisam de vários nós para cobrir uma pequena área.

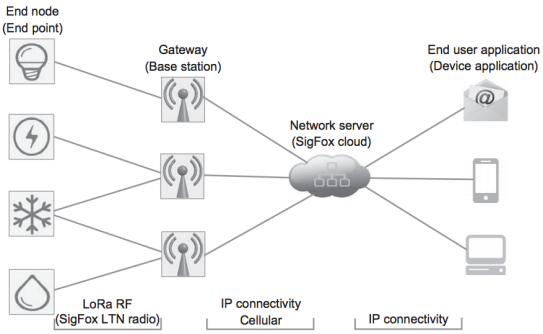

Figura 2.10 Arquitetura de rede LPWAN [16]

2.5.2 Camada física

Como visto anteriormente a camada física das redes 6LoWPAN é implementada pelo padrão IEEE802.15.4. Já as camadas físicas da LoRa e do SigFox são implementadas por dois outros padrões. A camada física da LoRa é chamada de LoRa RF que é uma técnica de espalhamento espectral, em específico Chirp Spread Spectrum, e tem como principais características o baixo custo, baixo consumo de energia e longo alcance [16]. Já a camada física do SigFox utiliza uma tecnologia de banda ultra estreita que possibilita uma rede em

grande escala, com alta capacidade e um consumo de energia muito baixo [16].

Um comparativo entre os parâmetros da camada física pode ser encontrado na Tabela 2.2.

Tabela 2.2 Parâmetros da camada física de 6LoWPAN, LoRaWAN e SigFox

Parâmetros	**6LoWPAN**	**LoRaWAN**	**SigFox**
Frequência	2400–2483.5 MHz (Mundo) 902–929 MHz (EUA) 868-868.6 MHz (Europa)	902-928 (EUA) 863-870 e 434 MHz (Europa) 779-787 MHz (China)	902 MHz (EUA) 868 MHz (Europa)
Canais	16 (2.4 GHz) 30 (915 MHz) 1 (868 MHz)	80 (915 MHz) 10 (780/868 MHz)	360 + 40 reservados
Banda	5 MHz (2,4 GHz) 2 MHz (915 MHz) 600 KHz (868 MHz)	125 KHz e 500 KHz (915 MHz) 125 KHz e 250 KHz (780/868 MHz)	100 Hz a 1,2 kHz
Taxa máxima	250 kbps	980 bps – 21,9 kbps (915 MHz) 250 – 50 kbps (780/868 MHz)	100 bps – 600 bps
Quadro	6 bytes header + 127 bytes SDU	# de bytes de header variável + 19 a 250 bytes	12 bytes header + 0 a 12 bytes
Modulação	O-QPSK	LoRa (915 MHz) LoRa e GFSK (780/868 MHz)	BPSK e GFSK
Uso do Espectro	DSSS	CSS	Ultra Narrow Band Coding
Sensibilidade	-85dBm (2,4 GHz) -92dBm (868/915 MHz)	-137 dBm	-137 dBm
Alcance	10 a 100 m	5 a 15 km	10 a 50 km
Tempo de vida	1 a 2 anos	Menos de 10 anos	Menos de 10 anos

Fonte: [5].

2.5.3 Camada de controle de acesso ao meio

Assim como na camada física a camada MAC também é definida pelo padrão IEEE802.15.4 nas redes 6LoWPAN. Em redes LoRa o protocolo tipicamente utilizado é chamado de

LoRaWAN e é um padrão aberto mantido pela LoRa Alliance [7].

O LoRaWAN separa seus dispositivos em três classes: A, B e C. Os dispositivos Classe A, a classe padrão, que é suportada por todos os dispositivos do protocolo, só recebe informações do *gateway* logo após uma transmissão e pode ficar em *sleep mode* por quanto tempo a aplicação definir. Os dispositivos Classe B além de receberem informações logo após a transmissão como os dispositivos da Classe A também possuem uma recepção síncrona. Já os da Classe C recebem informações continuamente exceto quando estão transmitindo.

Já o SigFox utiliza efetivamente utiliza o protocolo Random Unslotted Time-Frequency ALOHA. Quando um *host* envia uma mensagem à estação rádio base o dispositivo escolhe aleatoriamente canais não reservados para enviar a mensagem. A estação rádio base por sua vez escuta todos os canais para recuperar a mensagem [19].

2.5.4 Camadas superiores

Na seção anterior vimos que o 6LoWPAN é essencialmente uma camada de adaptação entre o protocolo IPv6 e o IEEE802.15.4 o que faz com que as suas principais funções sejam de comprimir, fragmentar e remontar os datagramas IPv6 para que os mesmos caibam dentro dos quadros IEE802.15.4 que são consideravelmente menores. Além disso outra função importante do 6LoWPAN é o roteamento feito pelo protocolo RPL para redes LLN.

Já as redes LPWAN não possuem suporte ao IPv6 e por isso não precisam de uma camada de adaptação. Em relação ao roteamento essas redes utilizam uma arquitetura estrela então o nó tem uma conexão ponto a ponto com o gateway, sendo desnecessário um protocolo de roteamento[16].

3 IMPLEMENTAÇÃO DA REDE 6LOWPAN

A implementação da rede 6LoWPAN realizada durante este trabalho utilizou dois tipos de plataformas de desenvolvimento. O Raspberry Pi 3 Model B para rodar o CETIC-6LBR que é um Border Router 6LoWPAN baseado no sistema operacional Contiki, e também dois kits de desenvolvimento LAUCHXL-CC1350 da Texas Instruments que serviram como interface entre o Border Router e a rede IEEE802.15.4 e o como o nó final, ambos rodando aplicações do Contiki.

3.1 HARDWARE

3.1.1 Raspberry Pi 3 Model B

O Raspberry Pi 3 Model B, Figura 3.3.1, é um modelo da plataforma de desenvolvimento Raspberry Pi desenvolvido pela Raspberry Pi Foundation que possui as seguintes configurações [20]:

a) Processador Quad Core 1.2GHz Broadcom BCM2837 64bit
b) 1GB RAM
c) 100 Mbps Fast Ethernet
d) 40 pinos para GPIO
e) 4 entradas USB 2.0
f) Entrada Micro SD para o Sistema Operacional e Armazenamento de dados
g) Alimentação através de Micro USB até 2.5A

Figura 3.3.1 Raspberry Pi 3 Model B [20]

3.1.2LAUNCHXL-CC1350

O kit de desenvolvimento CC1350, Figura 3.3.2, é um micro controlador sem fio de banda dupla (Sub-1 GHz e 2.4 GHz), onde foi utilizada a frequência de 915 MHz, e baixo consumo produzido pela Texas Instruments que possui, dentre outras, as seguintes características [21]:

a) Processador Arm Cortex - M3
b) Wireless M-Bus (EN 13757-4) e IEEE® 802.15.4g
c) Integrado com Code Composer Studio (CCS)

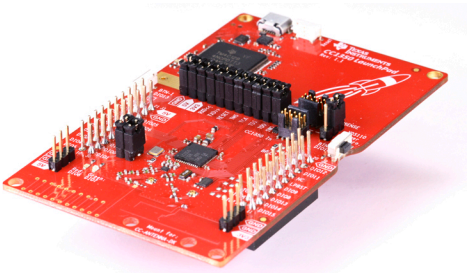

Figura 3.3.2 LAUCHXL-CC1350 [21]

3.2 SOFTWARE

3.2.1 CONTIKI OS

Contiki é um sistema operacional para a Internet das Coisas, fornecendo conexão à internet com baixo consumo de energia e dando suporte aos protocolos IPv6 e 6LoWPAN utilizados durante este trabalho [22].

3.2.2 CETIC-6LBR

CETIC-6LBR é uma aplicação baseada no Contiki que funciona como o Border Router da rede 6LoWPAN, fazendo a conexão entre a rede IEEE802.15.4 e a Internet [23], como na Figura 3.3.3.

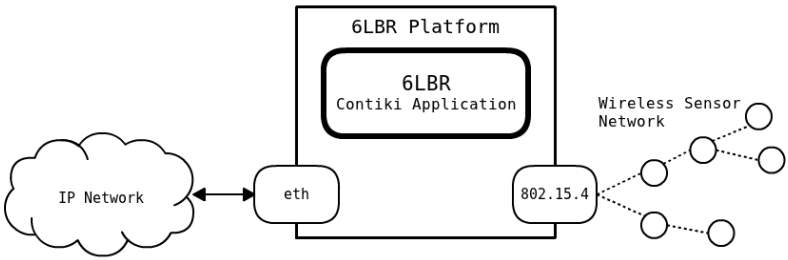

Figura 3.3.3 Exemplo da rede com a aplicação 6LBR [23]

3.2.3 RSMB

O RSMB, Really Small Message Broker, é um servidor do tipo *publish/subscribe* para os protocolos MQTT e MQTT-SN. A razão para utilizar o RSMB é que o principal servidor MQTT, o Mosquitto, não possui suporte ao MQTT-SN que é justamente o protocolo voltado a rede de sensores [24].

3.2.4 IoT MQTT Dashboard

É um aplicativo para celulares Android que funciona como um cliente e permite conectar ao RSMB e receber as informações dos sensores enviadas ao servidor, bem como enviar mensagens ao servidor para realizar o controle dos dispositivos [25].

3.3 IMPLEMENTAÇÃO

Esta seção apresenta um passo a passo de como adquirir, configurar e utilizar as ferramentas descritas acima para montar uma rede de sensores capaz de se comunicar com a internet.

3.3.1 Passo 1 – Instalando as ferramentas necessárias

Para que possamos rodar o Contiki precisamos do seu código fonte, que está disponível em [26], o CC1350 onde iremos gravar e executar os programas, além da *toolchain* necessária para compilar o Contiki para o CC1350. Em [27] é recomendada a toolchain GNU Tools for ARM Embedded Processors na versão 5.2.1 20151202. Já em [28] é possível encontrar como preparar o seu sistema operacional de escolha para compilar o Contiki; durante este trabalho foi utilizado o Linux Mint [29].

O restante dos drivers necessários para compilar e executar o Contiki são obtidos durante a instalação do CCS, disponível em [30]. No momento da instalação do CCS deve-se selecionar o suporte aos processadores SimpleLink CC13xx and CC26xx Wireless MCUs como mostra a Figura 3.3.4.

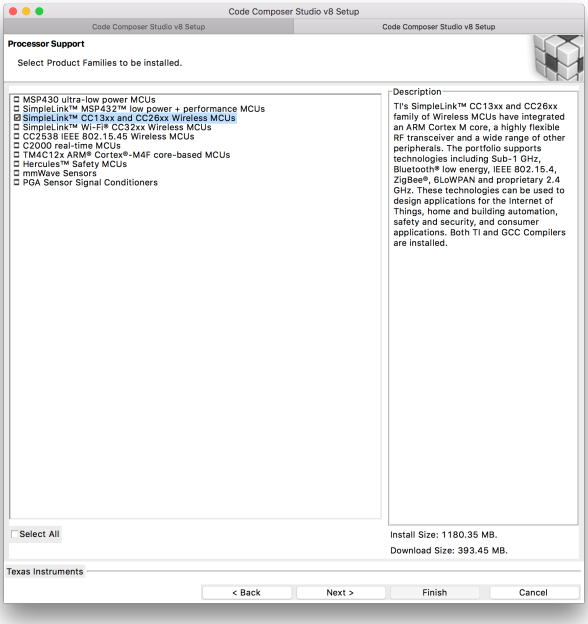

Figura 3.3.4 Instalação do CCS

3.3.2 Passo 2 – Configurando o CCS

Já com o CCS instalado e o ambiente de desenvolvimento devidamente preparado podemos configurar a IDE. Primeiro devemos importar o código fonte do Contiki como *"Existing Code as Makefile Project"* (Figura 3.3.5).

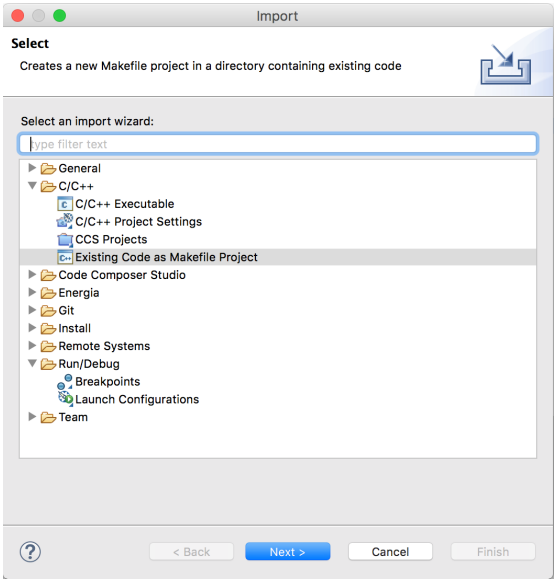

Figura 3.3.5 Importando o código fonte do Contiki

Na janela seguinte deve-se escolher como *toolchain* Cross GCC. Como mostra a Figura 3.3.6.

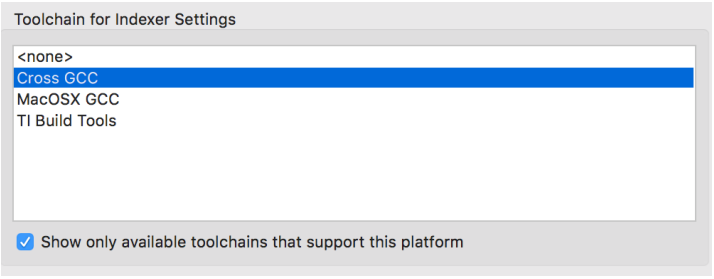

Figura 3.3.6 Selecionando a Toolchain

Agora que temos o código fonte do Contiki devidamente importado para o CCS devemos criar uma nova *Target Configuration* para que possamos gravar o programa no CC1350. Para isso vamos em File; New; Target Configuration File e configurá-lo de acordo com a Figura 3.3.7.

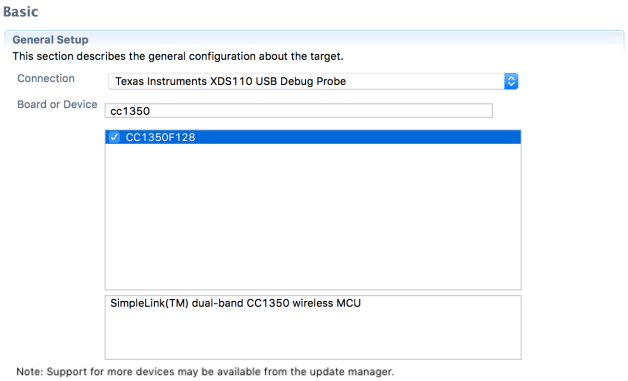

Figura 3.3.7 Target Configuration File

Depois de criado o arquivo de configuração do CC1350 devemos ir em Run; Debug Configuration e criar uma nova configuração em Code Composer Studio – Device Debugging utilizando esse arquivo como na Figura 3.3.8.

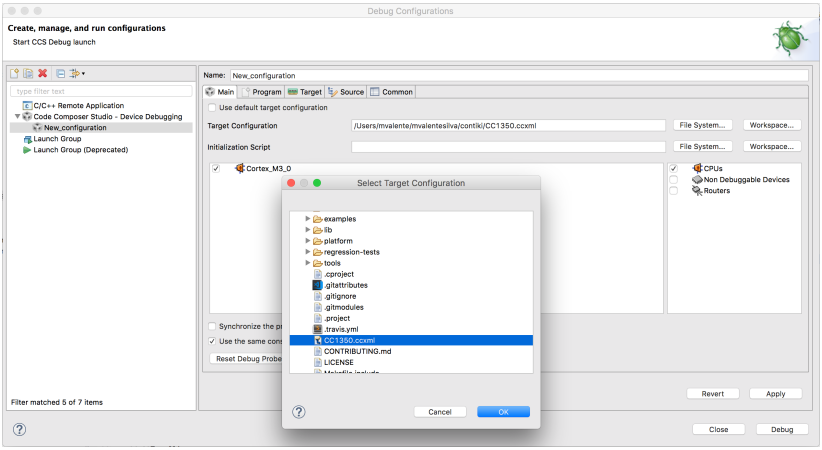

Figura 3.3.8 Debug Configuration

3.3.3 Passo 3 – MQTT-SN

Agora que o CCS está completamente configurado, vamos utilizar um cliente MQTT-SN para se comunicar com o servidor RSMB. Porém, bem como o servidor Mosquitto, o Contiki só tem suporte ao protocolo MQTT. O protocolo MQTT-SN, ou MQTT Sensor Network, foi desenvolvido para se adaptar às necessidades das redes de sensores sem fio, como pequena largura de banda, pequenas mensagens e alta taxa de falha de conexão. Além disso ele é otimizado para dispositivos de baixo custo, operados a bateria e com armazenamento e processamento limitados. A arquitetura de uma rede MQTT-SN pode ser vista na Figura 3.3.9. Mais informações sobre o protocolo MQTT-SN podem ser encontradas em [8].

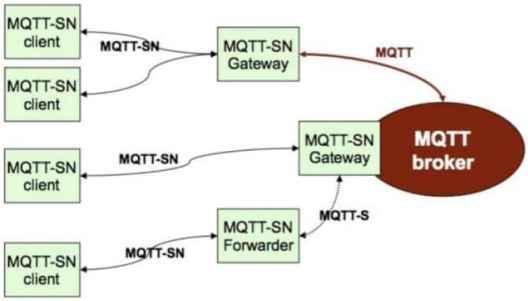

Figura 3.3.9 Rede MQTT-SN[24]

Com isso, vamos utilizar uma implementação disponível na internet sob a licença MIT [31], e com modificações realizadas pelos professores Guilherme Moritz e Ohara Rayel da Universidade Tecnológica Federal do Paraná disponível em [32].

Após obter o código fonte do cliente MQTT-SN e extraí-lo dentro do diretório de exemplos do Contiki, devemos criar um arquivo Makefile.target na pasta da aplicação para especificar a

plataforma e o dispositivo para qual desejamos compilar o programa com as especificações da Figura 3.3.10.

```
1 TARGET = srf06-cc26xx
2 BOARD = launchpad/cc1350
```

Figura 3.3.10 Arquivo Makefile.target

O arquivo **example.c** já apresenta um cliente MQTT-SN completamente funcional que se inscreve no tópico device_id/ctrl e publica periodicamente uma mensagem no tópico device_id/msg. Onde o device_id é o *rime address* do CC1350 sendo utilizado, que é dado pelos 64 bits menos significativos do IPv6 do nó. No caso deste trabalho o *id* é 00124B0013754400.

Para que o cliente seja capaz de se conectar ao servidor é necessário atualizar o IP ou o *hostname* do servidor como na Figura 3.3.1.1 e garantir que o valor da macro RESOLV_CONF_SUPPORTS_MDNS definida no arquivo resolv.h que está na em contiki/core/net/ip seja 1.

```
1 #define UDP_CONNECTION_ADDR 2801:84::1116:10C2:39ea:b970:4ca3
```

Figura 3.3.11 Atualização do arquivo example.c com o IP do servidor

Para que o cliente simulasse um sensor foi alterado o processo que enviava mensagens periodicamente para enviar números simulando um sensor de temperatura, assim podemos tirar vantagem da função de gráficos disponível no aplicativo IoT MQTT Dashboard. O processo que recebe as mensagens do tópico de controle foi alterado para que pudéssemos alterar os LEDs do CC1350 simulando um atuador. O código inteiro utilizado está disponível em [33].

Feitas as devidas alterações podemos compilar o programa e gravá-lo em um dos CC1350.

3.3.4 Passo 4 – Border Router

Neste passo vamos configurar o Raspberry Pi e instalar a aplicação CETIC 6LBR que servirá de interface entre a rede 802.15.4 (WPAN) e a Internet.

O Raspberry deve rodar a última versão do Raspian disponível em [34] e o acesso remoto deve estar habilitado seguindo os passos em [35]. Deve-se baixar e configurar o 6LBR seguindo o tutorial disponível em [36].

Primeiramente, devemos instalar um pacote que será necessário para configurar o modo bridge posteriormente, usando o seguinte comando:

```
sudo apt-get install bridge-utils
```

Depois, precisamos instalar as bibliotecas necessárias para compilar o 6LBR:

```
sudo apt-get install git build-essential libncurses5-dev
```

Com os pacotes anteriores devidamente instalados podemos fazer o download do código fonte do 6LBR:

```
git clone https://github.com/cetic/6lbr
cd 6lbr
git submodule update --init --recursive
cd examples/6lbr
```

Antes de compilar o 6LBR devemos definir o valor da seguinte macro como **CLOCK_SECOND**

```
#define SIXLBR_CONFIG_DEFAULT_SLIP_TIMEOUT (CLOCK_SECOND)
```

no seguinte arquivo

6lbr/examples/6lbr/platform/native/native-config.h

De acordo com [37], essa alteração corrige um erro quando se usa o 6LBR com o CC1350 como Slip Radio que será programado no próximo passo. Segundo a *thread* [37] é possível que o erro aconteça pelo fato do CC1350 utilizar uma *baudrate* menor que o padrão de outros dispositivos. Após corrigirmos essa diferença de taxa podemos compilar o 6LBR:

```
make all
make plugins
make tools
```

Agora instalaremos a aplicação:

```
sudo make install
sudo make plugins-install
sudo update-rc.d 6lbr defaults
```

Vamos então configurar o 6LBR para rodar no modo *Smart Bridge*, Figura 3.3.12. Esse modo permite a interconexão de uma rede padrão IPv6 com uma rede de sensores sem fio usando protocolo RPL [38], que por padrão é utilizado no Contiki. O RPL começa a identificar as rotas assim que a rede é iniciada [10].

Para isso vamos substituir os seguintes arquivos pelos fornecidos pelos professores Guilherme Moritz e Ohara Rayel da UTFPR:

a) /etc/6lbr/6lbr.conf
b) /etc/network/interfaces
c) /etc/dhcpcd.conf

Os arquivos acima podem ser encontrados nos ANEXOS A, B e C respectivamente. Além disso devemos habilitar a aceitação do prefixo de rede com os seguintes comandos na pasta /6lbr/examples/6lbr/tools:

```
sudo ./nvm_tool --new /etc/6lbr/nvm.dat
sudo ./nvm_tool --update --wsn-accept-ra 1 /etc/6lbr/nvm.dat
```

Além disso vamos também configurar id do canal da rede 802.15.4 como 25:

```
sudo ./nvm_tool --update --channel 25 /etc/6lbr/nvm.dat
```

Mais informações sobre as configurações da ferramenta nvm_tool podem ser encontradas em [39].

Com todas as configurações acimas realizadas, podemos reiniciar o Raspberry e inicializar o 6LBR através do comando:

```
sudo service 6lbr start
```

Também podemos observar o log de execução da aplicação com o seguinte comando:

```
sudo tailf /var/log/6lbr.log
```

3.3.5 Passo 5 – Slip Radio

Neste passo iremos gravar a aplicação Slip Radio do Contiki no segundo CC1350 para que ele funcione como a interface IEEE802.15.4 do 6LBR. Utilizando o CCS vamos compilar exemplo do Contiki que se encontra no seguinte diretório:

examples/ipv6/slip-radio

Como escolhemos o canal 25 para o 6LBR vamos definir o canal em **project-conf.h** da seguinte maneira:

```
#define RF_CORE_CONF_CHANNEL (25)
```

É importante lembrar que para compilar este programa também é necessário criar um arquivo Makefile.target da mesma maneira que fizemos no passo 3. Feito isso podemos compilar e gravar o programa no CC1350. Depois de gravado basta conectá-lo ao Raspberry executando o 6LBR e abrir o log de execução para observar a aplicação criando a rede.

3.3.6 Passo 6 – RSMB

Neste ponto a rede está praticamente pronta, o cliente configurado no passo 3 já foi reconhecido pelo *border router* configurado no passo anterior. Falta apenas instalar e executar o servidor.

O código fonte do RSMB pode ser obtido através de [24]. Em um Linux ou Mac, compilamos o programa:

```
cd rsmb/src
make
```

A seguir devemos criar um arquivo chamado **setup.cfg** com as configurações disponíveis em [40] para inicializar o servidor com o comando:

```
./broker_mqtts setup.cfg
```

3.3.7 Passo 7 – IoT MQTT Dashboard

A partir deste momento a rede já está funcionando normalmente. O cliente já se inscreveu no tópico ctrl e está enviando mensagens para o tópico msg. Agora podemos utilizar

o aplicativo IoT MQTT Dashboard para visualizar as mensagens e enviar sinais de controle ao cliente.

O próximo passo é criar uma nova conexão com as configurações da Figura 3.3.13, onde Client ID é um nome para o servidor, Server é o IP ou *hostname* e Port é a porta 1883 definida pelo arquivo **setup.cfg**.

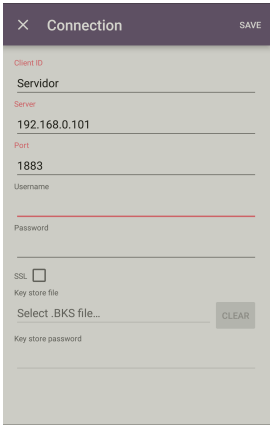

Figura 3.3.3.12 Nova conexão no IoT MQTT Dashboad

Depois podemos nos conectar ao servidor e definir os tópicos nos quais vamos inscrever o cliente para receber informações, e também em quais tópicos vamos publicar para fazer o controle dos LEDs do CC1350 rodando o cliente MQTT-SN. As configurações podem ser vistas na Figura 3.3.13.

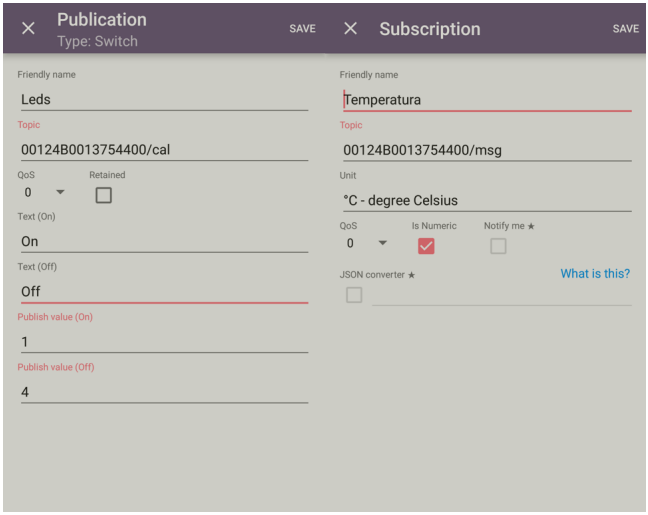

Figura 3.3.3.13 Configuração de Publish/Subscribe no IoT MQTT Dashboard

Com isso podemos controlar os LEDs do kit no Dashboard como na Figura 3.3.3.14.

Figura 3.3.3.3.14 Switch de controle dos LEDs

Além de observar as informações em tempo real como vemos na Figura 3.3.3.15

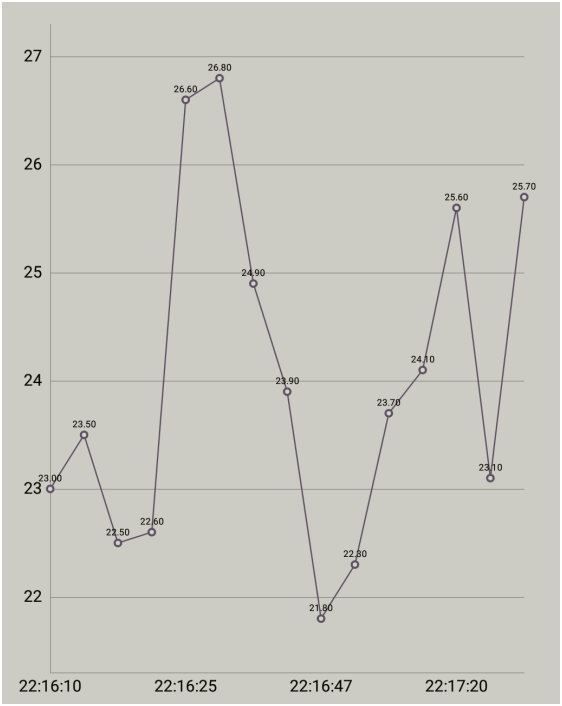

Figura 3.3.3.15 Gráfico em tempo real do tópico 00124B0013754400/msg

4 CONCLUSÃO

A utilização dos protocolos descritos ao longo deste trabalho apresenta soluções promissoras para o desenvolvimento das redes de sensores sem fio. A camada de adaptação 6LoWPAN permite a conexão de rádios de baixo consumo à internet através de IPv6 abrindo uma gama de possibilidades de aplicações.

Além disso, os softwares utilizados ao longo do projeto que contam com o código aberto, como o Contiki OS, 6LBR e o MQTT-SN, impulsionam o desenvolvimento de novas aplicações.

A implementação de uma rede de sensores com o o Contiki, o 6LBR e o MQTT-SN se mostrou relativamente fácil, onde as maiores barreiras foram as configurações dos computadores e o acesso a uma rede IPv6. Em trabalhos futuros seria interessante aproveitar a capacidade das placas de desenvolvimento disponíveis para a implementação de aplicações mais complexas.

REFERÊNCIAS

[1] NORDRUM, Amy. **Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated.** Disponível em: <https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated>. Acesso em: 17 nov. 2018.

[2] BANAFA, Ahmed. **Nine IoT Predictions for 2019.** Disponível em: <https://iot.ieee.org/newsletter/november-2018/nine-iot-predictions-for-2019>. Acesso em: 17 nov. 2018.

[3] KOCAKULAK, Mustafa; BUTUN, Ismail. An overview of Wireless Sensor Networks towards internet of things. **2017 Ieee 7th Annual Computing And Communication Workshop And Conference (ccwc)**, [s.l.], p.14-18, jan. 2017. IEEE. http://dx.doi.org/10.1109/ccwc.2017.7868374.

[4] LIBELIUM (Org.). **Smart water management for wastewater treatment in isolated communities.** Disponível em: <http://www.libelium.com/smart-water-management-for-wastewater-treatment-in-isolated-communities/>. Acesso em: 17 nov. 2018.

[5] SALMAN, N.; RASOOL, I.; KEMP, A. H.. Overview of the IEEE 802.15.4 standards family for Low Rate Wireless Personal Area Networks. **2010 7th International Symposium On Wireless Communication Systems**, [s.l.], p.701-705, set. 2010. IEEE. http://dx.doi.org/10.1109/iswcs.2010.5624516.

[6] LIBELIUM (Org.). **Reducing Logistics' environmental impact by air quality monitoring in the Baltic Sea Port of Gdansk, Poland.** Disponível em: <http://www.libelium.com/reducing-logistics-environmental-impact-by-air-quality-monitoring-in-the-baltic-sea-port-of-gdansk-poland/>. Acesso em: 17 nov. 2018.

[7] LIBELIUM (Org.). **LIBELIUM.** Disponível em: <http://www.libelium.com/>. Acesso em: 17 nov. 2018.

[8] MQTT.ORG (Org.). **MQTT Documentation.** Disponível em: <http://mqtt.org/documentation>. Acesso em: 15 nov. 2018.
[9] MA, Xin; LUO, Wei. The Analysis of 6LowPAN Technology. **2008 Ieee Pacific-asia Workshop On Computational Intelligence And Industrial Application**, [s.l.], p.963-966, dez. 2008. IEEE. http://dx.doi.org/10.1109/paciia.2008.72.
[10] COLINA, Antonio Liñán et al. **IoT in five days.** 1.1 [s.i]: [s.n.], 2016. 227 p. Disponível em: <https://github.com/marcozennaro/IPv6-WSN-book/releases/>. Acesso em: 15 nov. 2018.
[11] WIKIPEDIA. **OSI Model.** Disponível em: <https://en.wikipedia.org/wiki/OSI_model>. Acesso em: 20 nov. 2018.
[12] RISDIANTO, Aris Cahyadi; RUMANI, R.. IPv6 Tunnel Broker implementation and analysis for IPv6 and IPv4 interconnection. **2011 6th International Conference On Telecommunication Systems, Services, And Applications (tssa)**, [s.l.], p.139-144, out. 2011. IEEE. http://dx.doi.org/10.1109/tssa.2011.6095422.
[13] PALATTELLA, Maria Rita et al. Standardized Protocol Stack for the Internet of (Important) Things. **Ieee Communications Surveys & Tutorials**, [s.l.], v. 15, n. 3, p.1389-1406, 2013. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/surv.2012.111412.00158.
[14] RICHARDSON, Michael; ROBLES, Ines. **RPL- Routing over Low Power and Lossy Networks.** [s.i]: Ieft, 2015. 98 slides, color. Disponível em: <https://datatracker.ietf.org/meeting/94/materials/slides-94-rtgarea-1>. Acesso em: 20 nov. 2018.
[15] GOYAL, Mukul. **A Compression Format for RPL Control Messages.** Quebec, Cadaná: University Of Wisconsin Milwaukee, 2011. 30 slides, color. Disponível em:

<https://www.ietf.org/proceedings/81/slides/roll-6.pdf>. Acesso em: 20 nov. 2018.

[16] AL-KASHOASH, H. A. A.; KEMP, Andrew H.. Comparison of 6LoWPAN and LPWAN for the Internet of Things. **Australian Journal Of Electrical And Electronics Engineering**, [s.l.], v. 13, n. 4, p.268-274, out. 2016. Informa UK Limited. http://dx.doi.org/10.1080/1448837x.2017.1409920.

[17] SU, Jean Baptiste. **IoT World: Sigfox Sens'it Discovery To Boost Growth Of Connected Devices.** Disponível em: <https://www.forbes.com/sites/jeanbaptiste/2018/05/15/iot-world-sigfox-sensit-discovery-to-boost-growth-of-connected-devices/#2d7eae8f54c9>. Acesso em: 15 nov. 2018.

[18] LORA ALLIANCE (Org.). **LoRA Alliance.** Disponível em: <https://lora-alliance.org/>. Acesso em: 20 nov. 2018.

[19] FINNEGAN, Joseph; BROWN, Stephen. **A Comparative Survey of LPWA Networking.** 2018. Disponível em: <https://arxiv.org/abs/1802.04222>. Acesso em: 20 nov. 2018.

[20] THE RASPBERRY PI FOUNDATION (Org.). **Raspberry 3 Model B.** Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Acesso em: 15 nov. 2018.

[21] TEXAS INSTRUMENT (Org.). **SimpleLink™ Dual-Band CC1350 Wireless MCU LaunchPad Development Kit.** Disponível em: <http://www.ti.com/tool/LAUNCHXL-CC1350>. Acesso em: 15 nov. 2018.

[22] CONTIKI OS. **Contiki.** Disponível em: <http://contiki-os.org/>. Acesso em: 15 nov. 2018.

[23] CETIC (Org.). **6LBR.** Disponível em: <https://github.com/cetic/6lbr/wiki>. Acesso em: 23 out. 2018.

[24] ECLIPSE (Org.). **RSMB.** Disponível em: <https://github.com/eclipse/mosquitto.rsmb>. Acesso em: 15 nov. 2018.

[25] NGHIA TH (Org.). **IoT MQTT Dashboard.** Disponível em:

<https://play.google.com/store/apps/details?id=com.thn.iotmqttdashboard&hl=pt>. Acesso em: 15 nov. 2018.
[26] CONTIKI (Org.). **The official git repository for Contiki.** Disponível em: <https://github.com/contiki-os/contiki>. Acesso em: 15 nov. 2018.
[27] CONTIKI (Org.). **Getting Started with Contiki for TI CC26xx.** Disponível em: <https://github.com/contiki-os/contiki/blob/master/platform/srf06-cc26xx/README.md>. Acesso em: 15 nov. 2018.
[28] CONTIKI (Org.). **Contiki Wiki.** Disponível em: <https://github.com/contiki-os/contiki/wiki>. Acesso em: 15 nov. 2018.
[29] LINUX MARK INSTITUTE (Org.). **Linux Mint.** Disponível em: <https://linuxmint.com/>. Acesso em: 15 nov. 2018.
[30] TEXAS INSTRUMENT (Org.). **Code Composer Studio (CCS) Integrated Development Environment (IDE).** Disponível em: <http://www.ti.com/tool/CCSTUDIO>. Acesso em: 15 nov. 2018.
[31] OPEN SOURCE INICIATIVE (Org.). **MIT License.** Disponível em: <https://opensource.org/licenses/MIT>. Acesso em: 15 nov. 2018.
[32] HUMFREY, Nicholas et al. **Contiki adaptation of MQTT-SN-Tools (MQTT For Sensor Networks) protocol.** Disponível em: <https://github.com/utfprwsn/mqtt-sn-tools-contiki>. Acesso em: 15 nov. 2018.
[33] SILVA, Matheus Valente da et al. **Adaptações no cliente MQTT-SN.** Disponível em: <https://github.com/mvalentesilva/contiki/tree/master/examples/mqtt-sn-tools-contiki-master>. Acesso em: 15 nov. 2018.
[34] THE RASPBERRY PI FOUNDATION (Org.). **Raspbian.** Disponível em: <https://www.raspberrypi.org/downloads/raspbian/>. Acesso em: 15 nov. 2018.

[35] THE RASPBERRY PI FOUNDATION (Org.). **SSH (SECURE SHELL).** Disponível em: <https://www.raspberrypi.org/documentation/remote-access/ssh/>. Acesso em: 15 nov. 2018.

[36] CETIC (Org.). **Other Linux Software Configuration.** Disponível em: <https://github.com/cetic/6lbr/wiki/Other-Linux-Software-Configuration>. Acesso em: 15 nov. 2018.

[37] BAN-CO. **CC1310 web-demo ack timeout.** Disponível em: <https://github.com/contiki-os/contiki/issues/2030>. Acesso em: 15 nov. 2018.

[38] KHAN, Meer M. et al. Sink-to-Sink Coordination Framework Using RPL: Routing Protocol for Low Power and Lossy Networks. **Journal Of Sensors**, [s.l.], v. 2016, p.1-11, 2016. Hindawi Limited. http://dx.doi.org/10.1155/2016/2635429.

[39] CETIC (Org.). **Nvm Tool.** Disponível em: <https://github.com/cetic/6lbr/wiki/Nvm-Tool>. Acesso em: 15 nov. 2018.

[40] CETIC (Org.). **6LBR Modes.** Disponível em: <https://github.com/cetic/6lbr/wiki/6LBR-Modes>. Acesso em: 15 nov. 2018.

[41] MORE: Mecanismo online para referências, versão 2.0. Florianópolis: UFSC Rexlab, 2013. Disponível em: < http://www.more.ufsc.br/> . Acesso em: 15 nov. 2018.

ANEXO A – Arquivo /etc/6lbr/6lbr.conf

```
#####################################################################
#################
      #Logging configuration
      # These parameters configure the location of the log files
and the level of
      # logging

      #Location of log files, use - for standard output
      LOG_6LBR=/var/log
      LOG_6LBR_OUT=$LOG_6LBR/6lbr.log
      LOG_6LBR_ERR=$LOG_6LBR/6lbr.err


      #Logging level

      # 0 : FATAL  : Unrecoverable error detected
      # 1 : ERROR  : Error detected and handled by 6LBR
      # 2 : WARN   : Unexpected condition occurred or important
information message
      # 3 : INFO   : Runtime information
      # 4 : DEBUG  : Debug information
      # 5 : PACKET : Trace printed when a packet is sent/received
      # 6 : DUMP   : Actual packet sent/received
      # 7 : TRACE  : Debug traces at packet level

      LOG_LEVEL=4

      # Logging services : Log can be filtered according to the
service
      #See log-6lbr.h for the list of current services

      #LOG_SERVICES=ffffffff
```

ANEXO B – Arquivo /etc/network/interfaces

```
      auto br0
      iface br0 inet manual
          bridge_ports eth0
          bridge_stp off
          up                echo                0                >
/sys/devices/virtual/net/br0/bridge/multicast_snooping
          post-up ip link set br0 address `ip link show eth0 |
grep ether | awk '{print $2}'`
```

ANEXO C – Arquivo /etc/dhcpcd.conf

```
      # interfaces(5) file used by ifup(8) and ifdown(8)
      # Please note that this file is written to be used with
dhcpcd
      #  For  static  IP,  consult  /etc/dhcpcd.conf  and  'man
dhcpcd.conf'
      # Include files from /etc/network/interfaces.d:
      source-directory /etc/network/interfaces.d
      # A sample configuration for dhcpcd.
      # Allow users of this group to interact with dhcpcd via the
control socket.
      #controlgroup wheel
      # Inform the DHCP server of our hostname for DDNS.
      hostname
      # Use the hardware address of the interface for the Client
ID.
      clientid
      # Use the same DUID + IAID as set in DHCPv6 for DHCPv4
ClientID as per RFC4361.
      # Some non-RFC compliant DHCP servers do not reply with
this set.
      # In this case, comment out duid and enable clientid above.
      #duid
      # Persist interface configuration when dhcpcd exits.
      persistent
      #  Safe  to  enable  by  default  because  it  requires  the
equivalent option set
      # on the server to actually work.
      option rapid_commit
      # A list of options to request from the DHCP server.
      option  domain_name_servers,  domain_name,  domain_search,
host_name
      option classless_static_routes
      # Most distributions have NTP support.
      option ntp_servers
      # Respect the network MTU. This is applied to DHCP routes.
      option interface_mtu
      # A ServerID is required by RFC2131.
      require dhcp_server_identifier
      #  Generate  Stable  Private  IPv6  Addresses  instead  of
hardware based ones
      slaac private
      # Example static IP configuration:
      #interface eth0
      #static ip_address=192.168.0.10/24
      #static ip6_address=fd51:42f8:caae:d92e::ff/64
      #static routers=192.168.0.1
      #static      domain_name_servers=192.168.0.1      8.8.8.8
fd51:42f8:caae:d92e::1

      denyinterfaces eth0 tap0
```