UNIVERSIDADE FEDERAL DO ABC

ENGENHARIA DE INFORMAÇÃO

RODRIGO HIDEAKI MURAKAMI

INTERNET DAS COISAS: ESTUDO DE PROTOCOLOS E IMPLEMENTAÇÃO DE APLICAÇÕES

SANTO ANDRÉ

2017

RESUMO

O termo Internet das Coisas (Internet of Things) está associado a conexões de objetos físicos como dispositivos, aparelhos eletrodomésticos e veículos à Internet. Estes objetos possuem softwares, sensores e conectividade embarcados, permitindo a troca de informações entre eles. A Internet das Coisas recorre ao uso do método de identificação RFID (Radio-Frequency IDentification, do inglês, Identificação por radiofrequência) e de outras tecnologias sem fio como a IEEE 802.15.4, Bluetooth e Wi-Fi, representando o futuro da computação e comunicação. O objetivo deste projeto é o estudo de protocolos utilizados na Internet das Coisas, da implementação de algumas aplicações utilizando os sensores e das análises dos pacotes transmitidos por essas aplicações utilizando o Wireshark.

LISTA DE FIGURAS

Figura 1: Termostato Nest 10
Figura 2: Exemplo do Amazon Dash Button 10
Figura 3: Comparativo Web x Internet das Coisas 11
Figura 4: Quadro de slots (4 slots) e diagrama de transmissão e recepção 14
Figura 5: Topologia e cronograma 14
Figura 6: Rede IPv6 conectada a uma rede 6LoWPAN 17
Figura 7: Compressão dos cabeçalhos IPv6 no 6LoWPAN 18
Figura 8: Topologia RPL 20
Figura 9: Arquitetura CoAP 20
Figura 10: Quadro CoAP 21
Figura 11: O mote MTM-CM5000-MSP 24
Figura 12: Interface do Copper rodando no Mozilla Firefox 27
Figura 13: Esquema da rede 28
Figura 14: Interface do cliente CoAP e os serviços 29
Figura 15: O roteamento do roteador de borda 30
Figura 16: Representação da arquitetura REST 31
Figura 17: Roteador de borda inicializado 33
Figura 18: Conexão do mote à máquina virtual 35
Figura 19: Ativação do LED do mote 36
Figura 20: Informação sobre a carga de bateria do mote 36
Figura 21: Comparação de temperatura 37
Figura 22: Página Web com gráfico de temperatura interativo 38
Figura 23: Função Observe do sensor de temperatura 39
Figura 24: Interface gráfica do Wireshark durante a captura de pacotes 40

LISTA DE TABELAS

Tabela 1: Especificações do MTM-CM5000-MSP 26
Tabela 2: Temperaturas das 16h - 22h 37
Tabela 3: Informações de um pacote CoAP 41
Tabela 4: Captura do Método POST 42
Tabela 5: Captura do Método GET 42
Tabela 6: Captura da Função OBSERVE (GET e ACKNOWLEDGMENT) 44
Tabela 7: Recebimento dos Recursos Durante a Observação 45
Tabela 8: Fim da Observação de Recursos 45

SUMÁRIO

1 INTRODUÇÃO 6
2 INTERNET DAS COISAS 7
2.1 IEEE 802.15.4 - CAMADA PHY E MAC 12
2.3 RPL - ROUTING PROTOCOL FOR LOW-POWER AND LOSSY NETWORKS 19
2.4 COAP - CONSTRAINED APPLICATION PROTOCOL 20
2.5 CONTIKI OS 22
3 DESENVOLVIMENTO 24
3.1 HARDWARE 25
3.2 COPPER (Cu) 27
3.3 ESQUEMATIZAÇÃO DA REDE 27
3.2.1 CLIENTE COAP 28
3.2.2 ROTEADOR DE BORDA 29
3.2.3 SERVIDOR COAP 30
4 EXECUÇÃO DO PROJETO 32
4.1 EXEMPLOS DE APLICAÇÕES 35
4.1.1 ATUADOR DE ATIVAÇÃO DO LED 35
4.1.2 SENSOR DE CARGA DA BATERIA 36
4.1.3 SENSOR DE TEMPERATURA 37
4.1.4 OBSERVAÇÃO DE RECURSOS NO PROTOCOLO COAP 39
4.2 CAPTURA DE PACOTES COM O WIRESHARK 39
4.2.1 CAPTURA DO MÉTODO POST 41
4.2.2 CAPTURA DO MÉTODO GET 42
4.2.3 CAPTURA DA FUNÇÃO OBSERVE 43
5 CONCLUSÃO 46
6 REFERÊNCIAS BIBLIOGRÁFICAS 47

1 INTRODUÇÃO

Atualmente, mais de 3 bilhões de pessoas ao redor do mundo, isto é, aproximadamente, 45% da população mundial, [1] utilizam a Internet para navegar na web, receber e enviar mensagens, acessar conteúdos multimídia, para serviços, redes sociais e jogos. Nesse contexto, é previsível que dentre as próximas décadas, a Internet irá crescer ainda mais e conteúdos e serviços da rede estarão sempre disponíveis à nossa volta, proporcionando o surgimento de novos aplicativos, novos meios de trabalho, de interação, de entretenimento e de viver [2].

Nessa perspectiva, o conceito tradicional de Internet como uma rede de infraestrutura para os usuários finais vai enfraquecer e dar espaço para a interconexão de objetos “inteligentes” pervasivos no dia-a-dia. Assim, a atual infraestrutura da Internet dará o suporte necessário para a difusão e interconexão desses objetos [2].

A Internet das Coisas, do inglês *Internet of Things* (IoT), vem ganhando rapidamente um grande espaço nos cenários das telecomunicações. O conceito por trás da IoT é de que diversos objetos do nosso dia-a-dia, como carros e eletrodomésticos, aliados com Identificação por Radiofrequência (do inglês *Radio Frequency IDentification* - RFID), tags, sensores, atuadores e celulares e, através de esquemas diferenciados de endereçamento, podem interagir um com os outros e cooperar para atingirem um propósito comum. A principal força da IoT é o grande impacto em diversos aspectos da vida cotidiana e dos usuários [3].

Em 2008, o Conselho Nacional de Inteligência Nacional dos EUA (*U.S. National Intelligence Council* - NCI) mencionou que [4]:

> “Até o ano de 2025, nós de acesso à Internet residirão em coisas do cotidiano, pacotes de comida, móveis, documentos de papel, entre outros. O desenvolvimento atual aponta para futuras oportunidades e riscos que irão surgir quando pessoas puderem controlar remotamente, localizar e monitorar até os dispositivos mais mundanos e objetos. Demanda popular combinada com avanços tecnológicos pode resultar na difusão generalizada da Internet

das Coisas (IoT) que poderia, como a Internet atual, contribuir muito para o desenvolvimento econômico e a capacidade militar".

A fim de avaliar as características da Internet das coisas, o objetivo desse projeto é estudar os protocolos, realizar aplicações com motes (pequenos hardwares capazes de se conectar a Internet) e implementações da IoT em um hardware com o sistema operacional Contiki instalado. As aplicações utilizarão os sensores de temperatura, umidade e bateria. Por fim, as informações transmitidas e recebidas serão analisadas com o auxílio do software Wireshark.

2 INTERNET DAS COISAS

De um ponto de vista conceitual, a IoT se baseia em 3 princípios, relacionados a habilidade de um objeto inteligente: (1) a ser identificável (qualquer objeto possui uma identificação), (2) poder se comunicar (qualquer coisa é capaz de se comunicar) e (3) a interagir (qualquer coisa é capaz de interagir) [2]. O objeto inteligente tem as seguintes características:

- Possui um corpo físico e um conjunto de aspectos físicos (por exemplo, tamanho e formato);
- Possui funcionalidades que permite a comunicação como a habilidade de descobrir e aceitar mensagens e respondê-las;
- Possui um identificador único;
- Está associado a pelo menos um nome e um endereço. Sendo o nome algo legível ao ser humano e o endereço legível à maquina e utilizado para comunicação com o objeto;
- Possui capacidades básicas de computação;
- Pode possuir meios de sentir fenômenos físicos como temperatura, luz, nível de radiação eletromagnética ou ações de gatilho que possuem efeitos na realidade física (atuadores).

De uma perspectiva de nível de sistema, a Internet das Coisas pode ser vista com um sistema altamente dinâmico e distribuído, composto por um grande número de objetos inteligentes que produzem e consomem informações.

De uma perspectiva de nível de serviço, o principal problema está relacionado em como integrar as funcionalidades e recursos provenientes dos objetos inteligentes em serviços.

De uma perspectiva de usuário, a Internet das Coisas irá proporcionar uma quantidade grande de serviços sempre responsivos, que poderão auxiliar e dar suporte as atividades diárias dos usuários. O paradigma atual de serviços sempre ativos irá se transformar em sempre responsivos de acordo com as necessidades dos usuários [2].

A Internet das Coisas trará diversos benefícios ao conectar objetos do dia-a-dia à Internet. Algumas características e desafios estão listados a seguir:

- Heterogeneidade de dispositivos: a IoT será caracterizada por uma vasta heterogeneidade em termos de dispositivos, estes terão diferentes capacidades tanto do ponto de vista computacional quanto ao da comunicação, podendo resultar em problemas de compatibilidade;

- Escalabilidade: Com o crescimento de objetos que serão conectados globalmente à esta estrutura, haverão problemas relacionados à escalabilidade como em nomeação e endereçamento, devido ao tamanho da rede; gerenciamento de dados, devido à alta quantidade de objetos;

- Otimização de uso de energia: Muitos objetos deverão ser adaptados para que não seja necessária a substituição o recarregamento de baterias constante. Mesmo que possamos carregar celulares no nosso dia-a-dia com frequência, isso será impraticável com determinados dispositivos. Desse modo, é necessário um consumo médio reduzido e otimizado. Nem todos os objetos precisarão ser otimizados, porém, uma parcela significativa necessitará economizar energia;

- Interoperabilidade e gerenciamento de dados: Com a grande circulação de dados, será necessário a utilização de padrões para que as informações sejam transmitidas e utilizadas de forma eficiente para garantir a interoperabilidade. Porém, com a criação de novos dispositivos e padrões a cada dia, isto pode se tornar em um grande desafio;

- Segurança e privacidade: Como a IoT estará presente no dia-a-dia das pessoas de forma pervasiva, a segurança e a privacidade são aspectos que devem ser garantidos para que haja aceitação dos usuários [2];

- Protocolos: Protocolos de Internet existentes como o HTTP (*Hyper Text Transmission Protocol*) e o TCP (*Transmission Control Protocol*) não são otimizados para transmissões de baixo uso de energia devido aos cabeçalhos e meta-dados longos e extensos, dessa forma, será necessário a utilização de novos protocolos que levem em consideração as necessidades da IoT [5].

A Internet das Coisas não será responsável por criar novos dispositivos, mas sim incrementar sistemas já existentes com novas funcionalidades que interagem com o mundo físico.

Diversas aplicações da IoT nas mais diferentes áreas podem ser mencionadas, como por exemplo, nas áreas de gerenciamento, transportes, entretenimento, planejamento e gerenciamento urbano. Em relação ao gerenciamento urbano, podemos citar o exemplo da cidade de Padova na Itália, que com o auxílio da Internet das Coisas, foi possível desenvolver o conceito de uma Cidade Inteligente (*Smart City*), que tem como objetivo trazer benefícios paras os habitantes, melhorar o ambiente e a qualidade de serviços e facilitar o gerenciamento da cidade com o uso de tecnologia da informação e comunicação. Na cidade de Padova, foram instalados sensores de temperatura; sensores em postes de luz, para auxiliar na manutenção; sensores de benzeno, para monitorar a qualidade do ar e sensores de humidade [6].

Na aviação, a *Virgin Atlantic* comprou os aviões Boeing 787 que utilizam a Internet das Coisas que podem produzir até 0.5 *Terabytes* de dados por voo. Segundo o Diretor de Tecnologia da Informação da *Virgin Atlantic*, David Bull, desde os aviões até dispositivos de carga estarão conectados assim como cada peça do avião: motor, as asas e o trem de pouso. Se ocorrer algum problema no motor, eles saberão antes mesmo que ele aconteça [7].

O serviço de entregas UPS (*United Parcel Service*), começou a utilizar Internet das coisas para reduzir os impactos ao meio ambiente ao monitorar as distâncias das entregas, gastos de combustível e o motor dos veículos. O objetivo era aumentar a eficiência dos seus serviços e economizar dinheiro [8].

Um exemplo de aplicação domiciliar, é o termostato Nest (Figura 1). O Nest é facilmente instalado, ele se ajusta segundo as suas preferências anteriores e automaticamente se adapta a cada período do dia. Além disso, ele consegue

detectar se não há ninguém no ambiente e entra em modo ausente, economizando energia utilizando sensores internos e até mesmo a localização do seu telefone celular [9].

Figura 1: Termostato Nest [9].

Outro exemplo que pode fazer parte do cotidiano de uma pessoa comum é o Amazon Dash Button, que é um dispositivo com conexão Wi-Fi utilizado para realizar pedidos pela Amazon através de apenas o pressionar de um botão. Cada Amazon Dash Button representa um produto e ele possui um adesivo atrás permitindo a sua instalação em qualquer lugar que seja conveniente [10], como por exemplo na Figura 2 a seguir:

Figura 2: Exemplo do Amazon Dash Button [10].

O produto em questão é utilizado na lavagem de roupas, assim, logo que o usuário precisar mais desse produto, ele pode pressionar o botão, posicionado estrategicamente perto da máquina de lavar, e realizar um pedido. Uma notificação do pedido é enviada para um smartphone, permitindo assim o cancelamento, se necessário.

Assim, as aplicações da Internet das Coisas podem ser as mais diversas nos mais variados âmbitos, com o propósito de melhorar sistemas, economizar recursos e automatizar processos, por exemplo. Entretanto, ao passo que o desenvolvimento é grande e acelerado, novos problemas e desafios surgem, como o relacionado aos protocolos que são utilizados na Internet atualmente (XML, HTTP, TCP e o IPv6) que necessitam de adaptações para que sejam executados em dispositivos de baixo consumo de energia. Os protocolos já existentes impedem a evolução da Internet das Coisas, pois são extensos e ineficientes [11]. O quadro comparativo na Figura 3 a seguir lista algumas características dos protocolos atuais e os utilizados para a Internet das Coisas, sendo uma característica importante da IoT a eficiência no tamanho de cabeçalhos e conteúdos:

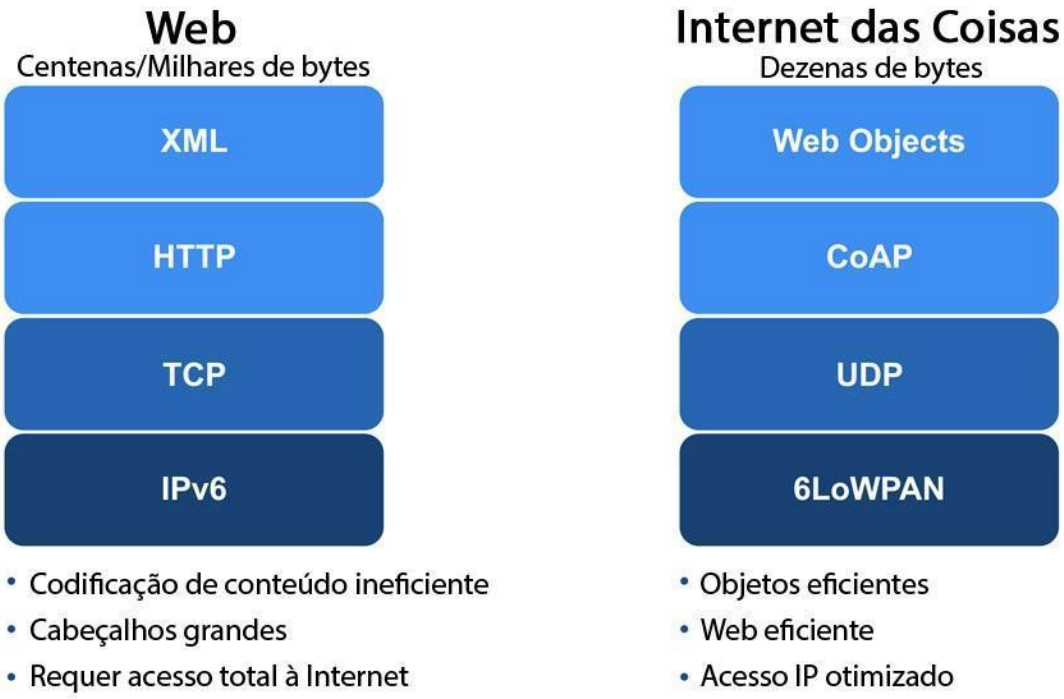

Figura 3: Comparativo Web x Internet das Coisas. Adaptado de [11]

Existem diversos protocolos criados que são proprietários, dificultando ainda mais a escalabilidade e o desenvolvimento da Internet das Coisas. Entretanto organizações como o IEEE (Instituto de Engenheiros Eletricistas e Eletrônicos) e o IETF (*Internet Engineering Task Force*) propuseram protocolos para os sistemas IoT [5].

Um dos protocolos mais importante foi o IEEE802.15.4, que define uma camada física de baixa potência (PHY) e um Controle de Acesso ao Meio (MAC, *Medium Access Control*), sendo utilizando em diversas tecnologias IoT [5]. Posteriormente, grupos de trabalho da IETF desenvolveram os protocolos 6LoWPAN (do inglês, *IPv6 over Low power Wireless Personal Area Networks*), que permitia a comunicação em redes pessoais sem fio de baixa potência utilizando o IPv6 [12]; ROLL RPL, um protocolo de roteamento e o CoAP. Todos esses avanços auxiliaram a comunicação sobre redes de baixa potência.

2.1 IEEE 802.15.4 - CAMADA PHY E MAC

Para que o desenvolvimento de objetos inteligentes que se comunicam da Internet das Coisas ocorresse, foi necessária a criação de uma camada PHY de baixa potência aliada com uma camada MAC que fosse eficiente no quesito de consumo de energia.

Geralmente, um hardware é equipado por uma bateria e trocá-la constantemente seria um trabalho custoso. Desse modo, para que a duração da bateria seja entendida é preciso que um hardware consuma pouca energia em média. A solução disso é a utilização de radio que consuma pouca corrente utilizando um protocolo mais eficiente que leve em consideração essas características.

O IEEE 802.15.4 PHY opera na faixa de frequência de 2.4 - 2.485 GHz e utiliza a modulação O-QPSK (*Offset-Quadrature Phase-Shift Keying* ou, em português, Modulação por Deslocamento de Fase com Atraso) com uma taxa de dados física de 2Mbps. Define 16 canais de frequência a cada 5MHz entre 2,405

GHz e 2,480 GHz, como os canais tem apenas 2 MHz de tamanho, eles não interferem um no outro.

O protocolo MAC presente no IEEE 802.15.4 define o formato do cabeçalho MAC (como os campos de fonte e destino) e como os objetos se comunicam entre si. Ela também prove a interface entre as camadas superiores e a camada PHY. O protocolo IEEE 802.15.4 MAC emprega o CSMA/CA (do inglês, *Carrier sense multiple access with collision avoidance*). O protocolo CSMA/CA é um mecanismo importante para o acesso de canais, levando em consideração a baixa taxa de transferência do IEEE 802.15.4. Esse mecanismo avalia o canal e permite os pacotes de dados a serem transmitidos caso a situação seja favorável, caso não seja, o algoritmo espera um determinado período de tempo até retransmitir.

A camada MAC é direcionada para redes do tipo estrela, em que todos os motes se comunicam diretamente com um nó coordenador [5]. Este protocolo é inadequado para redes multi-hop (redes em que a comunicação entre dois nós e enviada através de nós intermediários [13]), pois este protocolo opera apenas com um canal, o que resulta em instabilidades na rede, e caso cada objeto operasse como um roteador, o consumo de potência seria elevado.

Em 2010, foi introduzido no protocolo IEEE 802.15.4 o Time Synchronize Channel Hopping (TSCH), um protocolo MAC altamente confiável e de baixa potência, baseado no protocolo proprietário Tyme Synchronized Mesh Protocol (TSMP). Embora diferem em alguns aspectos, possuem as mesmas funções de sincronizar os nós para economizar energia e realiza o channel hop para confiabilidade.

No TSCH, os motes sincronizam em uma estrutura de quadro de slots que se repetem em um determinado intervalo de tempo. Cada objeto segue um cronograma que indica o que deve ser executado em cada slot. Em um slot, um objeto pode estar em estado de espera (“*sleep*”) ou ativo (que pode receber ou transmitir para um vizinho). No estado de *sleep*, o dispositivo não liga seu rádio e, em um estado ativo, o cronograma indica para qual vizinho receber ou transmitir.

Na Figura 4, pode-se perceber que em um quadro para o transmissor é possível apenas um pacote de comprimento máximo e receber uma mensagem de confirmação (*Acknowlodgment* - ACK) vinda do receptor.

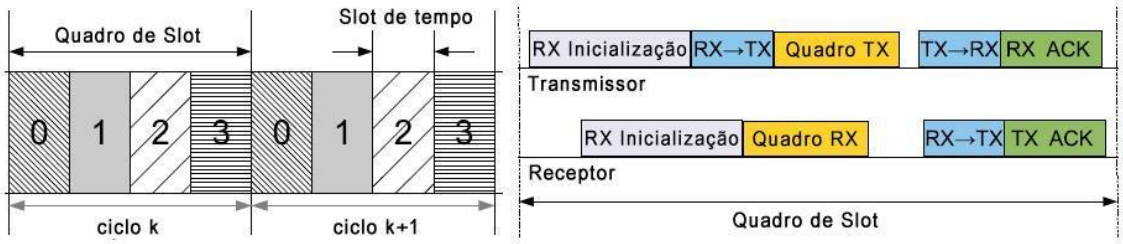

Figura 4: Quadro de slots (4 slots) e diagrama de transmissão e recepção [Adaptado de 5].

Um pacote gerado é enviado para a camada MAC que o coloca em uma fila de transmissão. A cada instante a camada MAC verifica se há um pacote destinado ao vizinho associado ao slot, caso haja, o pacote é transmitido e espera-se uma mensagem de confirmação ACK. Caso não haja, entra-se em modo *sleep* e desliga-se o rádio. A cada *slot* de recepção, um mote liga seu rádio e verifica se há algum pacote destinado a ele, se houver, ele recebe e envia uma confirmação ACK. Se não houver, desliga seu rádio e entra em modo *sleep*.

O cronograma para o envio de pacotes pode ser visto na Figura 5:

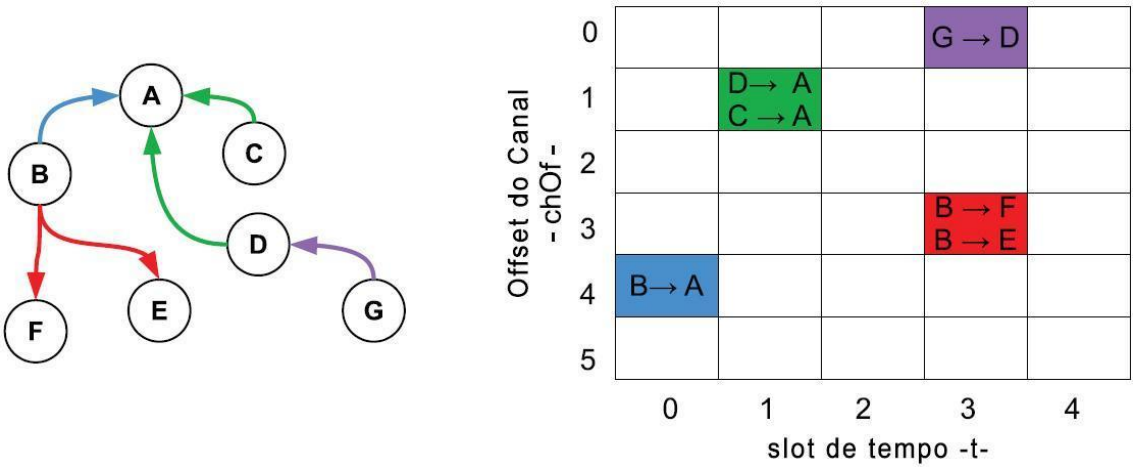

Figura 5: Topologia e cronograma [5].

A Figura 5 mostra a topologia e seu respectivo cronograma (do inglês, *schedule*). As ligações representam os vizinhos que cada mote pode transmitir. O quadro de *slots* possui 5 slots e 6 offsets de canal. Cada mote só se preocupa com a

célula que participa, logo, se B quiser enviar um pacote para F, irá esperar até o slot de tempo 3. Se um pacote precisa ser enviado de G para A, primeiro irá de G para D, sendo armazenado em D, e depois de D para A.

O protocolo IEEE802.15.4e define apenas como a camada MAC executa o cronograma para envio de pacotes. Como o cronograma é construído não está definido. Um cronograma deve ser desenvolvido com cuidado para que as transmissões sejam feitas de forma mais eficiente levando em conta a topologia. A criação do cronograma pode seguir duas linhas: centralizado ou distribuído. Em um cronograma centralizado, existe um administrador que é responsável por construir e gerenciar a rede. Possui informações sobre os motes, suas conexões. Quando há uma mudança, o administrador cria um novo cronograma e informa para os motes. Um sistema centralizado é eficiente, porém, quando a rede é sujeita a transformações constantes, um sistema distribuído pode ser mais interessante. Em uma abordagem distribuída, os motes decidem localmente a quem eles se conectam e seus cronogramas. Isso pode se tornar em um problema mais complexo caso cada mote gere dados a uma taxa constante, pois alguns desses hardwares podem não ser capazes de processar tantos dados de forma simultânea [5].

Para manter uma conectividade, é necessária a sincronização. Existem dois métodos de sincronização: baseado em *Acknowledgment* e baseado em Quadros. Na sincronização baseada em *Acknowledgment*, o receptor calcula o intervalo entre o tempo esperado da chegada do quadro e a chegada de fato e envia essa informação para o mote transmissor na mensagem de confirmação. Já na sincronização baseado em quadros, o receptor calcula o intervalo entre o tempo esperado da chegada do quadro e a chegada de fato e utiliza essa informação para ajustar seu clock.

O protocolo IEEE802.15.4e também adicional o *channel hopping* para o acesso de tempo de quadros. Além de aumentar a capacidade da rede, pois mais motes podem transmitir simultaneamente, reduz os efeitos da interferência e de desvanecimento de multi-caminhos.

2.2 IETF 6LOWPAN - IPV6 OVER LOW POWER WPAN

Devido à necessidade da criação de especificações para o transporte de pacotes IPs, o grupo IETF IPv6 over Low power WPAN (6LoWPAN) criou um protocolo para transmitir pacotes de IPv6 nas redes IEEE 802.15.4 [5].

Uma rede Low power WPAN tipicamente possui as seguintes características: pacotes de pequeno tamanho, largura de banda pequena, dispositivos alimentados com bateria, grande número de dispositivos, posições de nós desconhecidas e economia de energia a partir de longos períodos inativos.

O IETF teve grandes dificuldades para definir uma especificação efetiva, pois o IPv6 possui um tamanho grande com cabeçalhos extensos. Entretanto, sua adoção é necessária devido ao número estimado de objetos conectados à Internet no futuro. Estima-se que 30 bilhões de “coisas” estejam conectadas à Internet até o ano de 2020 [14]. Desse modo, a escalabilidade do IPv6 se torna essencial.

Para gerenciar um pacote de IPv6, utilizando encaminhamento e fragmentação, o protocolo 6LoWPAN usa uma camada intermediária entre o IPv6 e o IEEE 802.15.4 [15]. Os cabeçalhos do IPv6 também podem ser comprimidos quando descartando informações redundantes que podem ser inferidas de outras camadas [16].

Todos os datagramas encapsulados do 6LoWPAN possuem um cabeçalho. Estes cabeçalhos podem ser divididos por tipo, sendo eles:

- No 6LoWPAN *Header* (Cabeçalho de Não 6LoWPAN): utilizado quando um pacote não segue as especificações da 6LoWPAN devendo ser descartado;
- *Dispatch Header* (Cabeçalho de Expedição): utilizado para comprimir cabeçalhos do IPv6 ou gerenciar multicast/broadcast da camada de comunicação;
- *Mesh Addressing Header* (Cabeçalho de Endereçamento de Rede): permite os quadros do IEEE 802.15.4 serem encaminhados na camada de comunicação;
- *Fragmentation Header* (Cabeçalho de Fragmentação): utilizando quando um datagrama não cabe em um quadro do IEEE 802.15.4.

A compressão dos cabeçalhos do IPv6 no 6LoWPAN leva em consideração que estes campos serão pequenos, comuns ou fáceis de serem inferidos através de outras camadas, diminuindo a redundância no envio de pacotes.

Na Figura 6, pode-se observar uma rede utilizando o IPv6 e uma malha conectada que utiliza o 6LoWPAN:

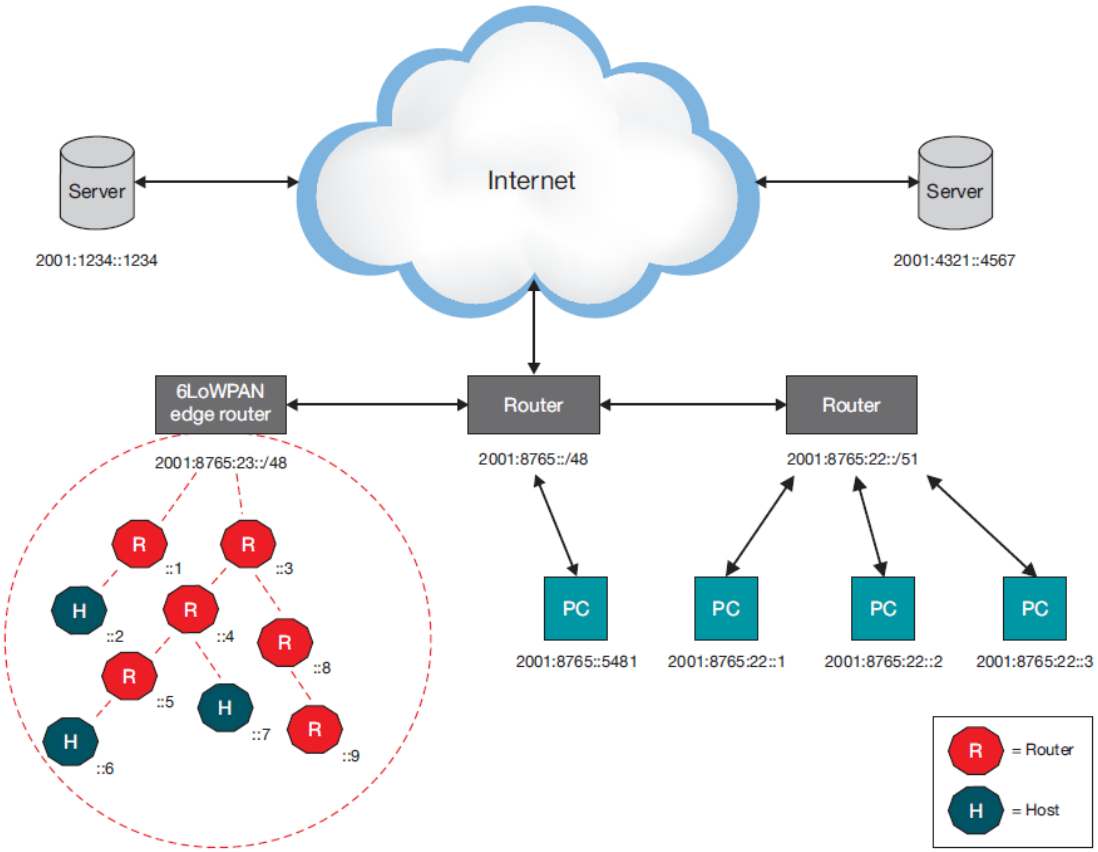

Figura 6: Rede IPv6 conectada a uma rede 6LoWPAN [17].

O acesso à Internet é garantido por um ponto de acesso (AP - Access Point) que é o roteador central IPv6. Nesta rede pode-se observar dispositivos conectados normalmente como computadores pessoais (PCs) e uma rede 6LoWPAN. A rede 6LoWPAN é conectada à rede IPv6 através de um roteador de borda. Esse roteador de borda executa as seguintes funções: (1) troca de dados entre os dispositivos da rede 6LoWPAN e a Internet; (2) troca de informações locais dentro da rede 6LoWPAN e (3) manutenção e gerenciamento da rede 6LoWPAN. Dessa forma, as redes 6LoWPAN comunicando-se através do protocolo IP, ela pode se conectar a outras redes que também utilizam esse protocolo [17].

Na Figura 7, encontra-se um exemplo da compressão dos cabeçalhos IPv6 no protocolo 6LoWPAN. Na parte superior encontra-se um cabeçalho IPv6 sem compressão e abaixo estão 3 cenários de compressão dos cabeçalhos.

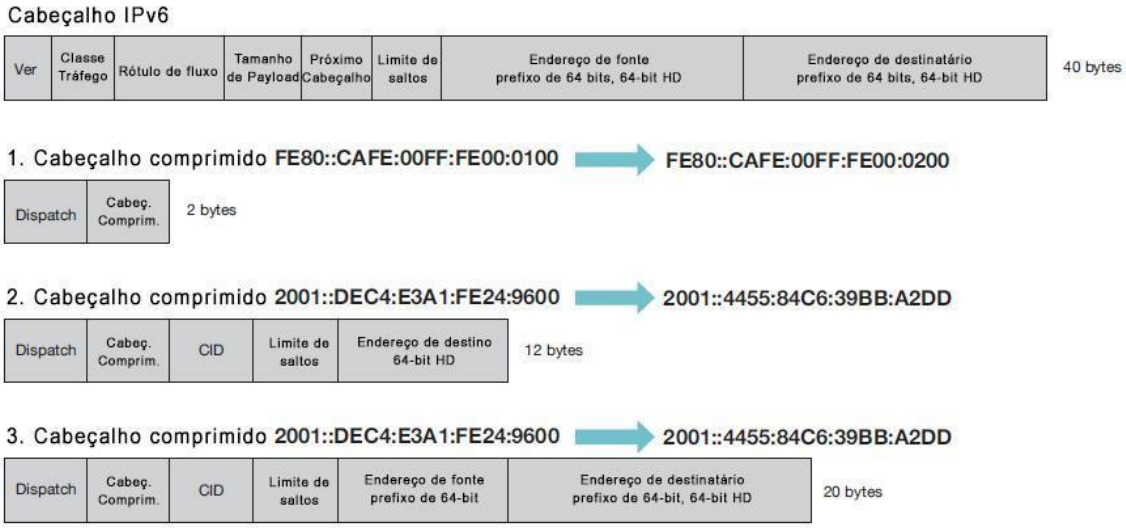

Figura 7: Compressão dos cabeçalhos IPv6 no 6LoWPAN [17].

O primeiro caso representa a comunicação de dois dispositivos em uma rede 6LoWPAN, utilizando um link local. O cabeçalho IPv6 pode então ser comprimido para 2 bytes.

No segundo caso, ocorre a comunicação de um dispositivo da rede 6LoWPAN para um dispositivo externo em que o prefixo da rede externa é conhecido. Assim, o cabeçalho IPv6 pode ser comprimido para 12 bytes.

No último caso, a comunicação também ocorre de modo similar ao segundo caso, entretanto, o prefixo da rede externa é desconhecido. O cabeçalho IPv6 é comprimido para 20 bytes.

Pode-se notar então que, até no pior caso, há uma compressão de 50% do cabeçalho IPv6.

2.3 RPL - ROUTING PROTOCOL FOR LOW-POWER AND LOSSY NETWORKS

Roteamento para o IPv6 é uma tarefa complicada devido às limitações como a baixa potência, comunicações de rádio com perdas, dispositivos alimentados a bateria. Por esses motivos, a IETF desenvolveu uma solução que foi o *IPv6 Routing Protocol* for *Low-power and Lossy Networks* (Protocolo de Roteamento IPv6 para Redes com Perda e de Baixa potência) - RPL.

O RPL dá suporte a diversas redes de baixa potência ou com perdas, ele é capaz de criar rapidamente rotas na rede e adaptar a topologia de um modo eficiente. Em uma arquitetura RPL típica, os nós da rede estão conectados através de caminhos com multi-saltos a um grupo de dispositivos raiz. Para cada nó é atribuido um *Destination Oriented Directed Acyclic Graph* (DODAG, Grafo Acíclico Orientado a Destino), atribuindo aos nós o custo de ligação, status, informação e uma função objetivo. A topologia é construída baseada em uma função objetivo que contém uma métrica ranqueada que leva em consideração a distância de cada nó em relação ao seu dispositivo raiz e restrições da rede [5].

Uma mensagem de controle utilizada pelo RPL é o DODAG *Information Object* (DIO - Objeto de Informação DODAG), que carrega informações de controle do roteamento, como por exemplo, o IPv6 do roteador raiz, posição do ranque de um nó. Outra mensagem de controle é a *Destination Advertisement Object* (DAO) que é utilizado para propagar informações sobre destino dentro da rede DODAG.

Em uma rede RPL existem três tipos principais de nós. O primeiro tipo são os nós raiz que providenciam a conectividade para outras redes; o segundo tipo são os roteadores que podem, por exemplo, enviar mensagens com a topologia para os vizinhos e, por último, são as folhas que não são capazes de enviar mensagens DIO, mas possuem a habilidade de participar de uma DODAG [18].

A construção das instâncias RPLs, que definem a função objetivo para uma DODAG, se baseiam em ranques. Existem métricas para definir as posições de cada nó, incluindo o uso da CPU, memória e energia disponível. Essa estruturação é crucial na formação de redes de baixa potência e com perdas. Por exemplo, na

Figura 8, o nó número 2, por estar mais próximo do nó de destino final, pode acabar ficando sobrecarregado. Por isso, o nó 7, mesmo não sendo a alternativa ótima, pode ser uma opção de roteamento dos dados.

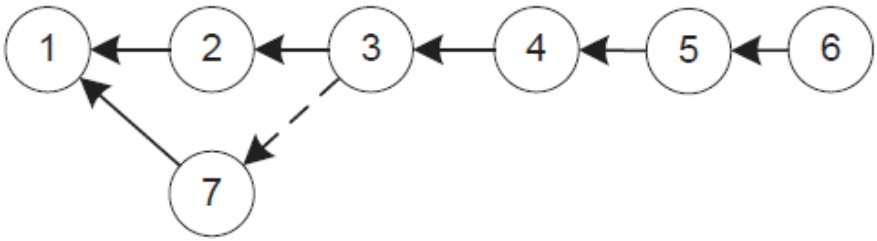

Figura 8: Topologia RPL [18]

O protocolo RPL foi desenvolvido para garantir uma comunicação confiável, com alta taxa de dados e, ao mesmo tempo, ter um consumo de energia eficiente em redes de baixa potência e com perdas.

2.4 COAP - CONSTRAINED APPLICATION PROTOCOL

O principal objetivo do protocolo CoAP é facilitar a compatibilidade e interoperabilidade entre as aplicações IoT e as já existentes, como por exemplo, o HTTP. Os protocolos de aplicações visam à interoperabilidade com outros protolocos já existentes sem que haja a necessidade de códigos orientados e especializados para aplicações. O grupo IETF definiu o CoAP que se traduz facilmente para o HTTP facilitando a integração com a Web. Ao invés de apenas comprimir o HTTP, o grupo definiu especificações REST (*Representational State Transfer* ou Transferência de Estado Representacional) permitindo a interoperabilidade com o HTTP ao mesmo tempo atendendo os requerimentos de protocolos com restrições especializados para a Internet das Coisas [5].

A Figura 9 representa a arquitetura CoAP:

Figura 9: Arquitetura CoAP [5].

Pode-se notar que o protocolo CoAP é dividido em duas camadas, sendo elas a de Requisição/Resposta e a de Mensagem. Fazendo-se uma analogia com a arquitetura usual OSI, o CoAP estaria situado entre a camada de aplicação e a de transporte. Outra alteração ocorrida foi a da utilização do protocolo UDP - User Datagram Protocol, pois a quantidade de informação necessária para realizar o controle e ter confiabilidade no protocolo TCP não seria uma opção viável devido às limitações.

A função da camada de Mensagens é controlar a troca de mensagens sobre o UDP entre dois pontos. As mensagens são identificadas por um ID que é utilizado para detectar mensagens duplicadas e para a confiabilidade [5].

A camada de Requisição/Resposta é responsável pelo controle de requisições e respostas. Por exemplo, se um servidor não for capaz de responder uma requisição em um determinado momento, ele responde através de uma mensagem de confirmação em branco para que o transmissor parar de tentar retransmitir a mensagem [19].

O cabeçalho de um quadro CoAP possui um tamanho fixo e ele pode ser visto na Figura 10:

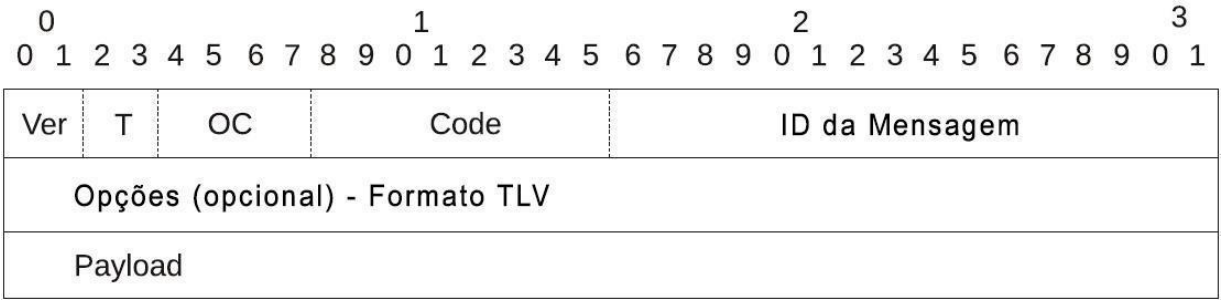

Figura 10: Quadro CoAP [5].

Os campos do quadro CoAP são:

- Ver: Versão, 2 bits representando o número da versão CoAP;
- T: Tipo, 2 bits para indicar o tipo de mensagem;
- OC: Contador Opcional, 4 bits que indicam o número de opções no cabeçalho de oções;
- Código: 8 bits indicando se a mensagem se trata de uma requisição ou uma resposta;
- ID da Mensagem: 16 bits, um campo que contém a identificação única da mensagem;
- Opções: lista de opções no formato TLV;
- *Payload*: contem é conteúdo da mensagem.

Os métodos básicos do CoAP são:

- *GET*: Método utilizado para receber ou requisitar uma informação;
- *POST*: Requisita um processamento de alguma representação no CoAP, geralmente um novo recurso ou um recurso ser atualizado;
- *PUT*: Requisita a atualização ou criação de um recurso;
- *DELETE*: Este método deleta um recurso.

2.5 CONTIKI OS

Contiki é um sistema operacional de código aberto desenvolvido para a Internet das Coisas, conectando micro hardwares de baixo custo à Internet. Ele suporta os protocolos IPv6, IPv4 e os padrões da IoT: 6LoWPAN, RPL e CoAP [20].

Nesse trabalho foi utilizado o Contiki OS devido às diversas facilidades que ele proporciona quando se trata do desenvolvimento, simulação e teste em aplicações de Internet das Coisas. O Contiki OS é disponibilizado como uma imagem chamada de Instant Contiki que pode ser rodada em programas de máquina virtual como o VMWare. O Instant Contiki é um ambiente completo de desenvolvimento. É uma máquina virtual que é executada no Ubuntu Linux, pode ser rodado no software VMWare Player e possui todas as ferramentas de desenvolvimento, simulação, compiladores e softwares já instalados [21].

Algumas das características que podem ser mencionadas do Contiki OS são:

- Alocação de memória: como o Contiki foi desenvolvido para rodar em sistemas pequenos com baixa capacidade de armazenamento, o sistema utiliza-a de modo altamente eficiente, possuindo alocador de memória;
- Suporte completo a redes IP: o Contiki fornece suporte aos protocolos já conhecidos como o IP, UDP, TCP e HTTP, além dos protocolos de baixa potência como os 6lowpan, RPL e o CoAP;
- Gerenciamento de energia: o sistema foi desenvolvido levando em consideração os hardwares de baixa potência, operando com o mínimo uso de bateria. Provendo mecanismos que permitem estimar e controlar o consumo de bateria;
- Simulador Cooja: o simulador de redes do Contiki permite o desenvolvimento, teste, simulação e depuração de redes sem fio de forma rápida e detalhada antes de implantá-las no mundo real;
- Prothothreads: é um mecanismo de baixa sobrecarga para programação concorrente utilizado pelo Contiki para economizar memória e ainda prover um fluxo contínuo na execução de códigos;
- Interpretador de códigos do Contiki: o Contiki possui um interpretador de linhas de comandos que são muito úteis durante o desenvolvimento. Esses códigos podem ser combinados com os já existentes do sistema Unix e os desenvolvidos por aplicações.

O Contiki consegue ser rodado em um grande gama de hardwares e foi desenvolvido para se adaptar facilmente a novos, consumindo pouca memória e

bateria. Designado para utilizar protocolos já conhecidos como o IPv4, IPv6 e HTTP, de fácil utilização devido a simplicidade do Instant Contiki e, além disso, é Opensource, permitindo a edição e sua utilização de forma comercial e não-comercial [21].

3 DESENVOLVIMENTO

O desenvolvimento do projeto iniciou com a instalação do VMWare e do Instant Contiki 2.7. A implantação desse trabalho se baseou na utilização de 2 motes rodando o Contiki OS e com os padrões e protocolos, discutidos nas seções anteriores, para demonstrar algumas aplicações da Internet das Coisas. Os motes utilizados foram o Tmote Sky, um deles atuou como sensor (servidor CoAP) e o outro como roteador de borda (cliente CoAP).

O Instant Contiki é um ambiente de desenvolvimento, simulação e teste completo com todas as ferramentas necessárias para a Internet das Coisas. O ambiente pode ser rodado em uma máquina virtual, como o VMWare, e já vem pronto para iniciar o desenvolvimento de aplicações. Como o desenvolvimento desse ambiente é baseado no sistema UNIX, existem muitos comandos já conhecidos e outros do Contiki OS disponíveis e úteis na programação para a Internet das Coisas. Uma ferramenta muito interessante presente no sistema Instant Contiki é o Cooja, que é um simulador em que é possível configurar diversos motes e rodar programas desenvolvidos para testá-los antes. Após a simulação, é possível realizar o upload do programa para os motes.

Inicialmente, um dos problemas encontrados foi em um dos motes utilizados para ser o servidor CoAP. O roteador de borda não conseguia localizar as rotas para se conectar ao servidor. Ao trocar de mote para ser o servidor, esse problema foi resolvido. Outro problema comum foi ao habilitar diversos sensores ao mesmo tempo, a memória do mote não conseguia comportá-los, isso foi resolvido ao desabilitar alguns sensores e recursos.

O ambiente utilizado para o desenvolvimento nesse projeto foi o Contiki OS. A linguagem de programação utilizada foi C e os programas desenvolvidos foram

baseados em aplicações que estão disponíveis no ambiente. A seguir, descrevem-se os métodos e o hardware empregados para esse trabalho.

3.1 HARDWARE

Os hardwares utilizados como mote são do modelo MTM-CM5000-MSP, produzidos pela empresa Advanticsys, totalmente compatível com a norma IEEE 802.15.4 e baseado na plataforma opensource TelosB/Tmote Sky, ele atua na faixa de 2,4 a 2,485 GHz, com sensibilidade de recepção de -95 dBm e um alcande de aproximadamente 120 metros (em um ambiente externo) e de 20-30 metros (em um ambiente fechado). Os sensores incluídos de fábrica são capazes de medir a temperatura, umidade relativa do ar e luz [22]. O mote possui as seguintes características:

- Plataforma WSN (Wireless Sensor Network) IEEE 802.15.4;
- Processador TI MSP430, CC2420 RF;
- Compatível com TinyOS 2.x e ContikiOS;
- Sensores de temperatura, umidade e luz
- Botões de usuário e reset;
- 3 LEDs;
- Interface USB;
- Compartimento para 2 baterias AA [23].

O MTM-CM5000-MSP pode ser observado na Figura 11 e suas especificações, na Tabela 1.

Figura 11: O mote MTM-CM5000-MSP [22].

Tabela 1: Especificações do MTM-CM5000-MSP [22]

Item	Especificação	Descrição
Processador		
Modelo do Processador	Texas Instruments® MSP430F1611	Texas Instruments® Família do MSP430
Memória	48KB 10KB 1MB	Flash do programa Data RAM Flash Externo (ST® M25P80)
ADC	Resolução de 12bit	8 canais
Interfaces	UART, SPI, I2C USB	Interfaces Seriais Interface de Sistema Externa (FTI® FT232BM)
Rádio		
RF Chip	Texas Instruments® CC2420	Módulo Wireless IEEE 802.15.4 2.4GHz
Frequency Band	2.4GHz ~ 2.485GHz	Compatível com o IEEE 802.15.4
Sensibilidade	-95dBm typ	Sensibilidade de recepção
Taxa de Transferência	250Kbps	Compatível com o IEEE 802.15.4
Potência RF	-25dBm ~ 0dBm	Configurável via software
Distância	~120m(externa), 20~30m(interna)	Distâncias mais longas são possíveis com a adição de outras antenas
Current Draw	RX: 18.8mA TX: 17.4mA Sleep mode: 1uA	Modos RF de potência reduzidas diminuem o consumo
RF Power Supply	2.1V ~ 3.6V	Potência de entrada CC2420
Sensores		
Luz 1	Hamamatsu® S1087 Series	Distância visível (560 nm de sensitivity wavelength)
Luz 2	Hamamatsu® S1087 Series	Visible & Infrared Range (960 nm peak sensitivity wavelength)
Temperatura e Umidade	Sensirion® SHT11	Faixa de Temperatura: -40 ~ 123.8 °C Resolução de Temperatura: ± 0.01 (típica) Precisão de Temperatura: ± 0.4 °C (típica) Faixa de Umidade: 0 ~ 100% RH Resolução de Umidade: 0.05 (típica) Precisão de Umidade: ± 3 %RH (típica)
Características eletromecânicas		
Dimensões	81.90mm x 32.50mm x 6.55mm	Includindo conector USB
Peso	17.7g	Sem bateria
Potência	3V (2xAA)	Regulador de potência MICREL® MIC5207

3.2 COPPER (Cu)

A aplicação cliente utilizada para receber os dados do servidor CoAP foi a extensão para Mozilla Firefox chamada Copper (Cu). O Copper é uma ferramenta de gerenciamento acessível para uma rede de dispositivos embarcados de Internet das Coisas, possibilitando uma interação de uma forma amigável e intuitiva [24]. Através dessa extensão é possível acessar serviços programados nos motes servidores CoAP. A interface do Copper pode ser observada a seguir:

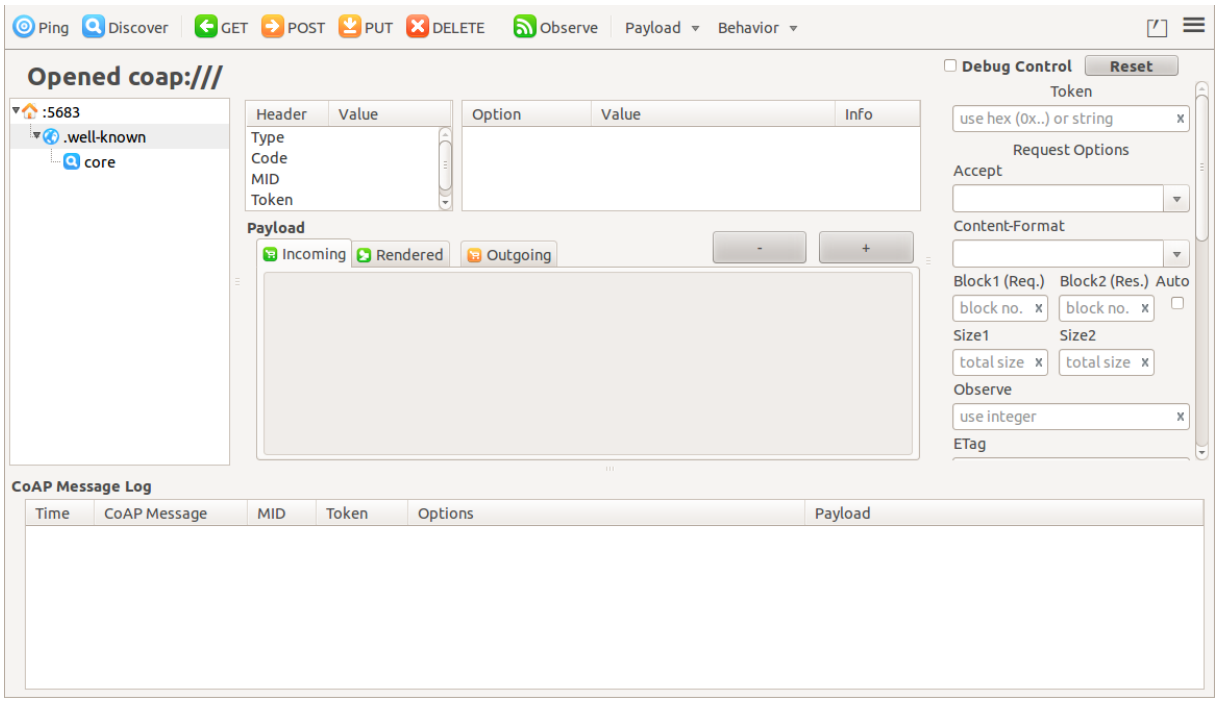

Figura 12: Interface do Copper rodando no Mozilla Firefox.

3.3 ESQUEMATIZAÇÃO DA REDE

A esquematização da rede utilizada nesse projeto foi estruturada utilizando três principais componentes: o servidor CoAP, o roteador de borda e um cliente CoAP. A estrutura pode ser observada na Figura 13 a seguir.

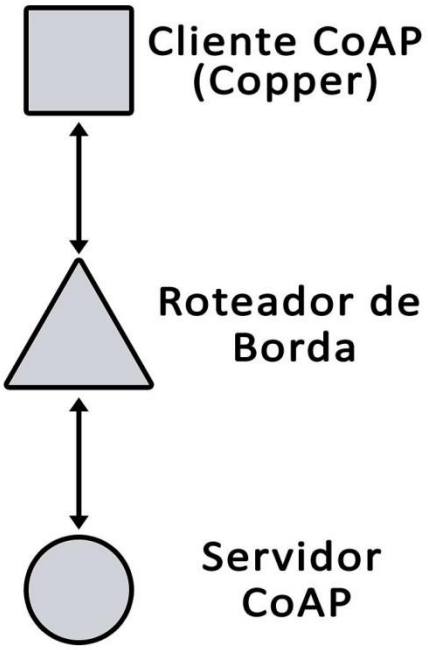

Figura 13: Esquema da rede.

A comunicação ocorre em todas as direções seja entre o cliente e o roteador e o roteador e o servidor, todos podem tanto enviar quanto receber dados. O roteador de borda acaba sendo um intermediário no envio de informações do servidor até o usuário.

3.2.1 CLIENTE COAP

O cliente CoAP nesse projeto foi um computador rodando a extensão Copper (Cu) no navegado Mozilla Firefox. O Copper permite o recebimento de informações do servidor CoAP e os mostra para o usuário em uma interface amigável e de fácil interação. Desse modo, é possível acessar todos os serviços disponíveis no servidor CoAP. Na Figura 14, que mostra a interface e os serviços (sensores) do servidor, podem-se verificar os botões: Ping, Discover, GET, POST, PUT, DELETE e Observe. A lista de serviços habilitados se encontra do lado esquerdo. Para acessar os serviços foi necessário obter o endereço do IPv6 do servidor CoAP através do roteador de borda. O endereço do servidor no caso era 212:7400:16c0:5107. Para

utilizar o Copper, a URL acessada foi: coap://[aaaa::212:7400:16c0:5107]:5683/, sendo 5683 a porta.

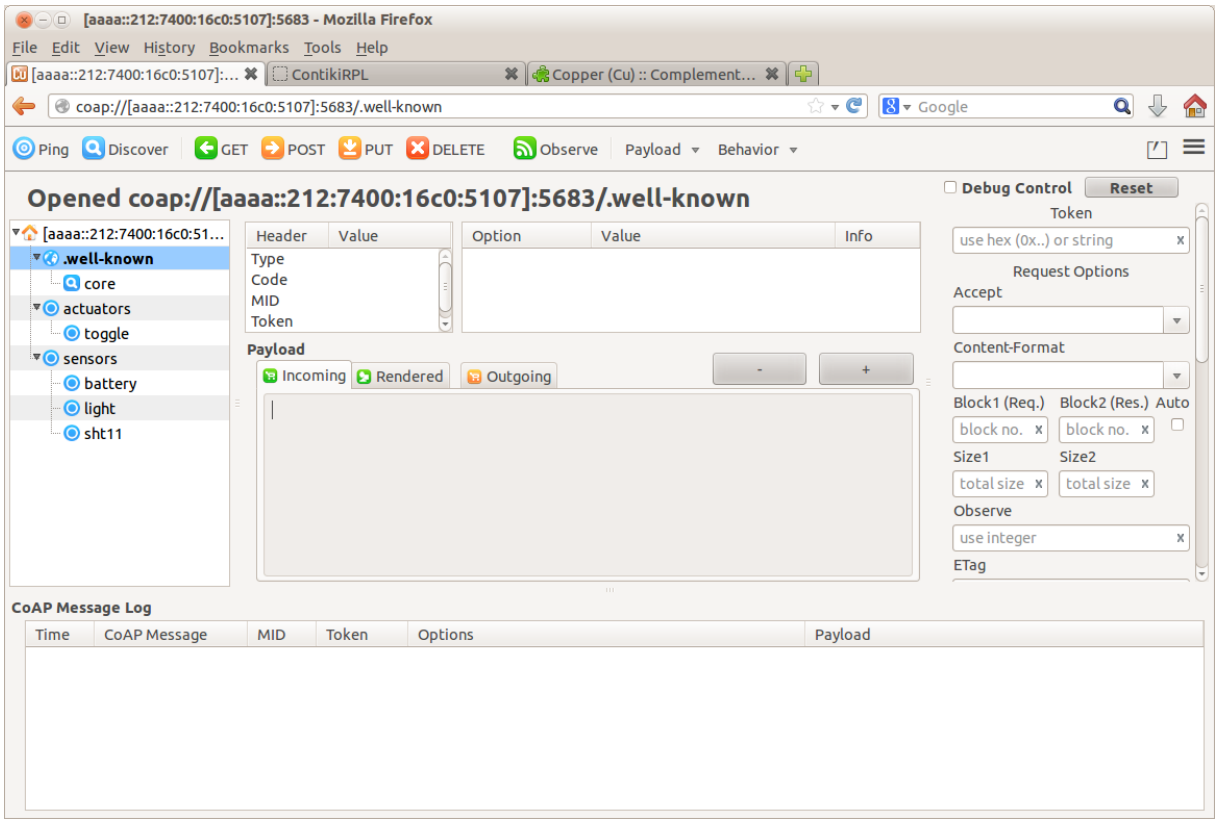

Figura 14: Interface do cliente CoAP e os serviços.

3.2.2 ROTEADOR DE BORDA

O roteador de borda é utilizado com o intuito de criar a interface entre uma rede convencional IP a uma rede RPL 6LoWPAN e costuma estar localizado na ponta da rede. O roteador utilizado nesse projeto se encontra no diretório do Instant Contiki: contiki/examples/ipv6/rpl-border-router [25]. O roteador também é responsável por descobrir motes novos que são adicionados a rede e de atribuir rotas à eles. A partir do nó do roteador de borda, é criada uma ponte que se conecta a rede RPL. Assim é possível ver essa conexão na Figura 15.

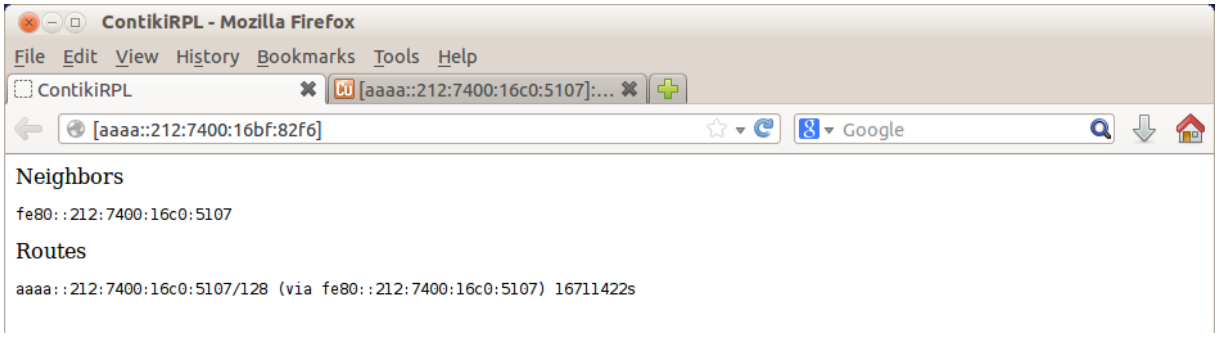

Figura 15: O roteamento do roteador de borda.

Essa figura demonstra que o roteador de borda conseguiu localizar e se conectar ao mote com sucesso e, a partir disso, foi possível acessar os serviços do servidor CoAP. O roteador possui um vizinho fe80::212:7400:16c0:5107 de endereço IPv6 aaaa::212:7400:16c0:5107.

3.2.3 SERVIDOR COAP

Os servidores CoAP nesse projeto possuíam os serviços que poderiam ser acessados pelo usuário através da conexão entre os servidores, o roteador de borda e o cliente. O mote possui alguns sensores que já vem de fábrica, sendo eles o de temperatura, umidade e luminosidade, estes que foram utilizados. O desenvolvimento da aplicação que foi rodada no mote foi baseado em um dos exemplos presentes no Instant Contiki no diretório: contiki/examples/er-rest-server.

O servidor utilizado é do tipo REST, do inglês Representational State Transfer, que é um estilo de arquitetura que consiste em um conjunto de restrições arquiteturais coordenadas aplicadas aos seus elementos e componentes. Na Figura 16, pode-se notar que no centro da arquitetura se encontram os recursos do REST, que são identificados por URI (Uniform Resource Identifiers) e que possuem uma representação interna. Por fim, há um conjunto de operações (POST, GET, PUT e DELETE) que define como esses dados serão utilizados [26]. O REST define uma arquitetura cliente-servidor em que o cliente tem acesso somente a uma representação dos recursos, eles não são acessados diretamente.

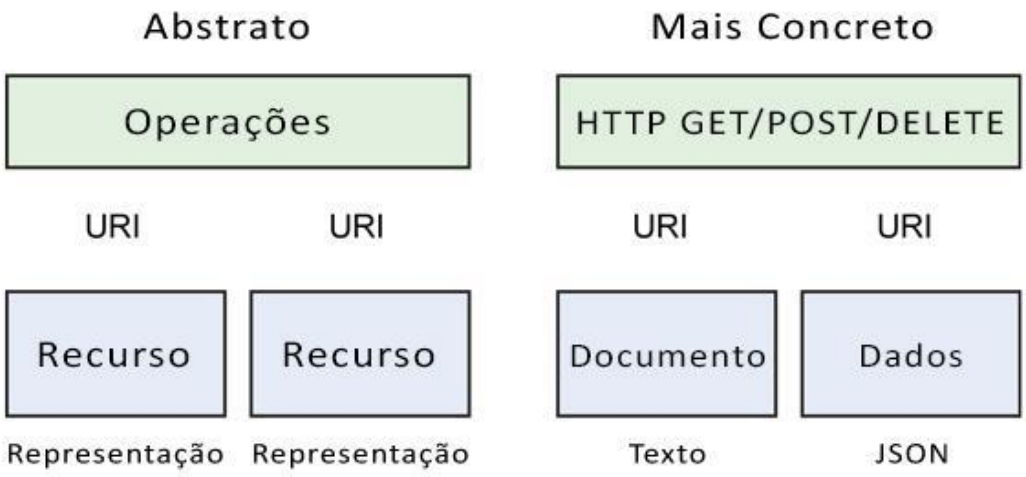

Figura 16: Representação da arquitetura REST [27].

No programa é possível escolher quais sensores que o servidor disponibiliza e como serão apresentadas as saídas, caso existam. O servidor fica em modo de espera até que chegue uma requisição de um cliente. Quando ela chega, o servidor então processa, devolve uma resposta e retorna ao modo de repouso. O código desenvolvido para o servidor disponibilizava os seguintes serviços:

- Atuador para acender LED: o cliente envia um comando para o servidor para que o LED seja aceso, possibilitando a sua identificação em uma rede de diversos motes;
- Sensor de bateria: que permite receber a carga da bateria;
- Sensor de luminosidade: serviço que fornece a luminosidade do ambiente em Lux;
- Sensor de temperatura: através do sensor SHT11, é possível obter a temperatura do ambiente em graus ºC;
- Sensor de umidade: através do sensor SHT11, é possível obter a umidade relativa do ambiente.

Um problema observado nessa parte de adição de serviços ao mote foi da limitação de memória que acabava impedindo que mais serviços fossem ativados e executados.

4 EXECUÇÃO DO PROJETO

Primeiramente, foi instalado o software VMWare Workstation Player que permite rodar sistemas operacionais e aplicações em uma máquina virtual, no caso, o Instant Contiki. Assim, para gravar o roteador de borda no mote, foram executados os seguintes passos:

1. Conectar o primeiro mote no USB do computador e habilitá-lo no Instant Contiki (Figura 18);
2. Abrir o Terminal no Instant Contiki;
3. Executar o comando de troca de diretório para o caminho do roteador de borda:

   cd contiki/examples/ipv6/rpl-border-router
4. Configurar o tipo de mote em que o Contiki será executado, no caso, um mote TmoteSky:

   make TARGET=sky savetarget
5. Compilar o código e realizar o upload no mote com o seguinte comando:

   make border-router.upload

Para realizar o upload do servidor CoAP, foram seguidos os passos a seguir:

1. Conectar o segundo mote no USB do computador;
2. Abrir um segundo Terminal no Instant Contiki;
3. Executar o comando de troca de diretório para o caminho do roteador de borda:

   cd contiki/examples/er-rest-example/
4. Configurar o tipo de mote em que o Contiki será executado, no caso, um mote TmoteSky:

   make TARGET=sky savetarget
5. Compilar o código e realizar o upload no mote indicando o número dele com o seguinte comando:

   make er-example-server.upload MOTE=2
6. Retirar o segundo mote com o servidor CoAP da entrada USB e inserir duas pilhas AA em seu compartimento de bateria.

Para iniciar a conexão entre o roteador de borda e o servidor CoAP, foram realizadas as seguintes etapas:

1. No terminal em que foi realizado o upload do roteador de borda, executar o comando de conexão:

make connect-router

2. Uma tela similar como a Figura 17, deverá aparecer, indicando que a conexão foi iniciada:

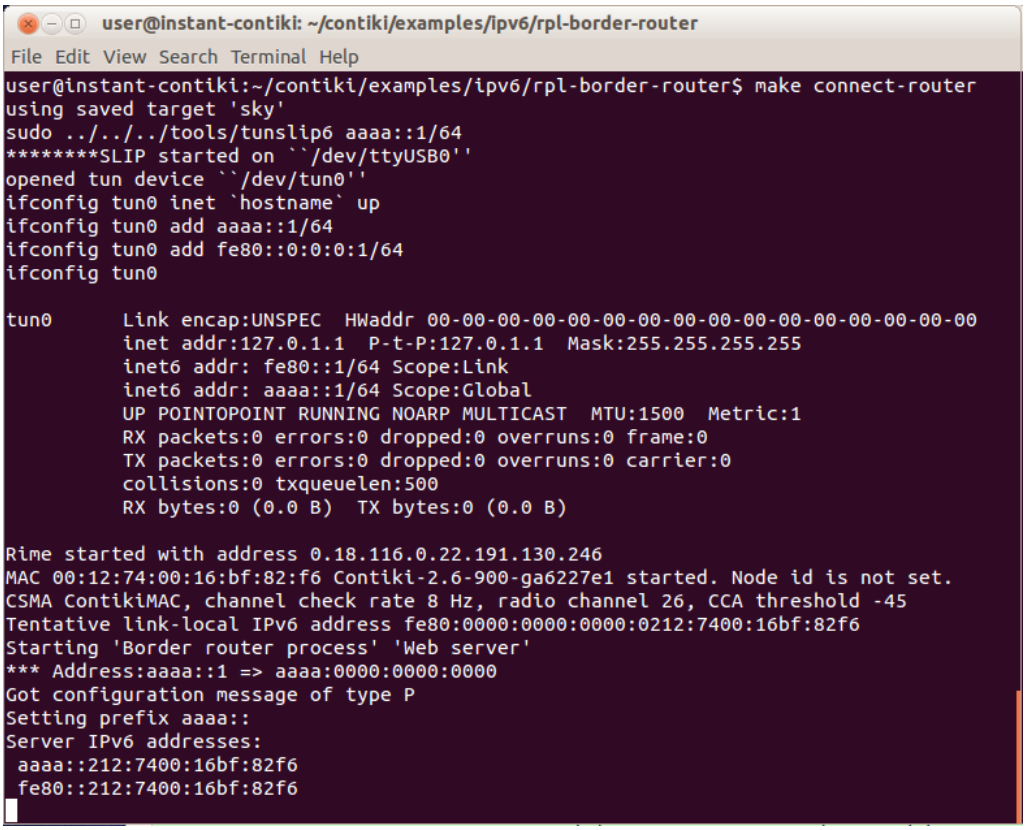

Figura 17: Roteador de borda inicializado.

Pode-se notar que o endereço IPv6 do roteador de borda desse exemplo foi:

aaaa::212:7400:16bf:82f6

Para acessar os serviços do servidor CoAP:

1. Abrir o Mozilla Firefox dentro do Instant Contiki;
2. Na barra de endereços, digitar o endereço do IPv6 do roteador de borda entre chaves:

[aaaa::212:7400:16bf:82f6]

3. Uma tela como é mostrada na Figura 15 deve aparecer com os endereços dos servidores CoAP, com as seguintes informações:

   Neighbors

   fe80::212:7400:16c0:5107

   Routes

   aaaa::212:7400:16c0:5107/128

   (via fe80::212:7400:16c0:5107) 16711421s

4. Com o endereço da rota, pode-se acessar o cliente CoAP Copper, através da url:

   coap://[aaaa::212:7400:16c0:5107]:5683/

5. Apertar o botão Discover para que os serviços sejam descobertos pelo cliente.
6. Os serviços já podem ser acessados através da seleção da barra lateral e pressionando os botões, como GET e POST na parte superior do Copper.

Observações:

1. Caso não seja possível realizar o upload no mote e a seguinte mensagem apareça:

   could not open port: [Errno 13] Permission denied: '/dev/ttyUSB0'

   Adicionar o usuário ao grupo dialout com o seguinte comando:

   sudo gpasswd --add ${USER} dialout

   Após a execução do comando, efetuar o logout e tentar o upload novamente;
2. Quando conectar o mote na porta USB, verificar se o hardware está conectado a máquina virtual como indicado na Figura 18.
3. Caso a mensagem a seguir apareça:

   region `rom' overflowed by 10 bytes

   collect2: ld returned 1 exit status

Ela demonstra que a memória do mote foi excedida e alguns recursos devem ser desativados para que o upload ocorra com sucesso.

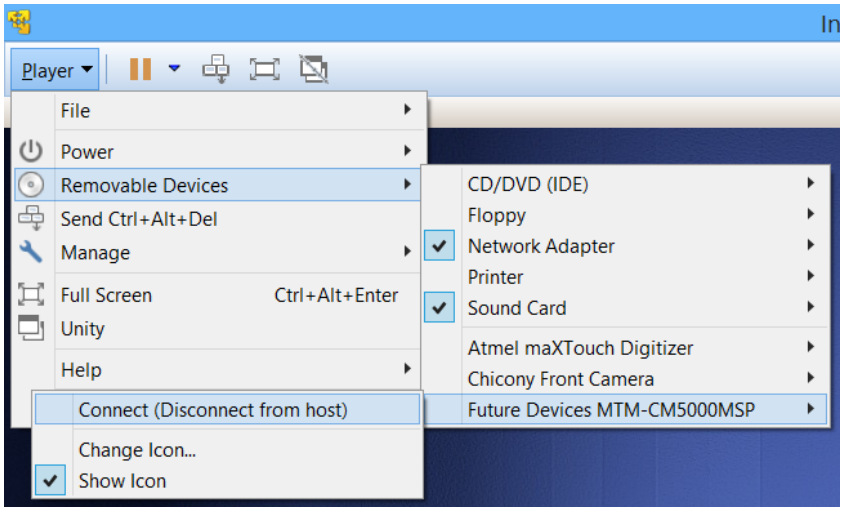

Figura 18: Conexão do mote à máquina virtual.

4.1 EXEMPLOS DE APLICAÇÕES

A seguir, encontram-se alguns exemplos de aplicações do servidor CoAP desenvolvido, incluindo o da proposta desse trabalho, a página Web exibindo os dados do sensor de temperatura.

4.1.1 ATUADOR DE ATIVAÇÃO DO LED

Um exemplo simples, porém muito útil, em grandes redes RPL para identificar os motes. Através do serviço de atuador, foi possível enviar um comando POST do cliente para o servidor CoAP para que ele acendesse um de seus LEDs. A Figura 19 demonstra esse funcionamento:

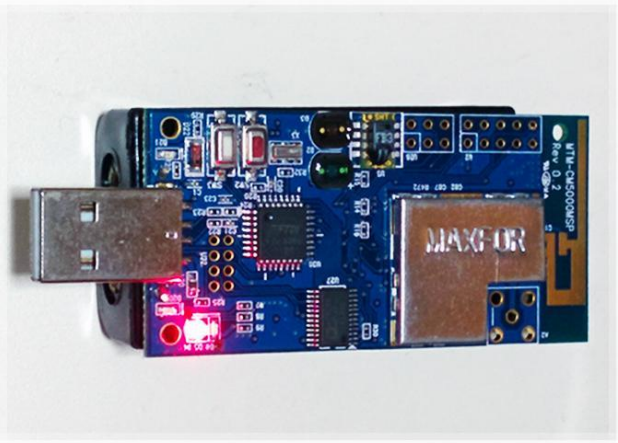

Figura 19: Ativação do LED do mote.

4.1.2 SENSOR DE CARGA DA BATERIA

O gasto de bateria é um quesito crítico em hardwares utilizados para Internet das Coisas. A bateria deve durar por um grande intervalo de tempo para que não seja necessário a sua reposição com muita frequência, por esse motivo, seu monitoramento é interessante para a análise do seu consumo. Um serviço no servidor foi programado para esse propósito. Através da operação GET, foi possível receber a porcentagem de carga da bateria, como pode ser visto na Figura 20.

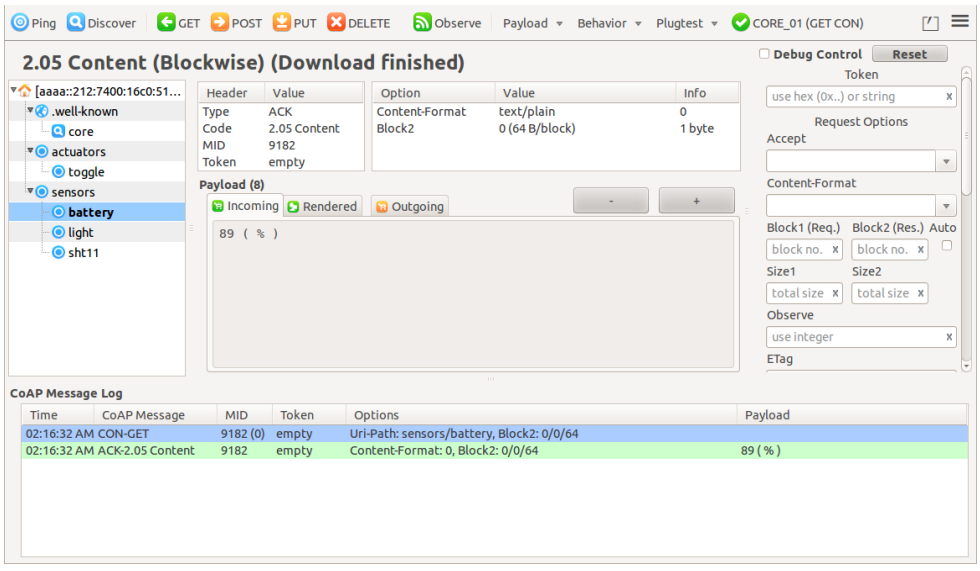

Figura 20: Informação sobre a carga de bateria do mote.

4.1.3 SENSOR DE TEMPERATURA

Utilizando o sensor presente no mote que agia como servidor CoAP, foram medidas 7 temperaturas ao longo de 6 horas, com o mote perto de uma janela. A medida inicial foi às 16h, e a primeira temperatura obtida no mote foi de 24 ºC. Comparando essa medida com o site ClimaTempo.com.br, pode-se validar o valor obtido da temperatura pelo sensor.

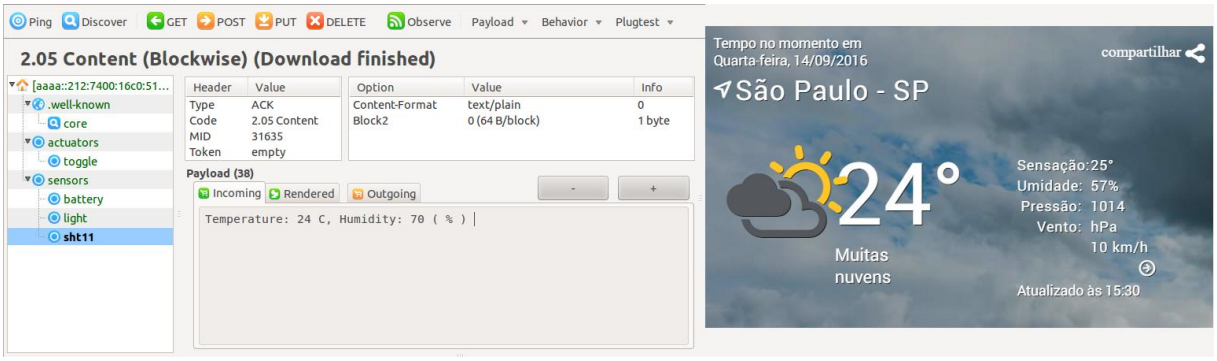

Figura 21: Comparação de temperatura.

Foram tomadas 7 medidas por hora que se encontram na Tabela 2.

Tabela 2: Temperaturas das 16h - 22h

Hora	Temperatura (ºC)
16:00	24
17:00	24
18:00	24
19:00	23
20:00	23
21:00	23
22:00	21

Uma página foi desenvolvida utilizando as tecnologias HTML, Javascript e Bootstrap para exibir os dados em um gráfico de uma forma mais organizada, interativa e amigável. Além disso, foi utilizado uma biblioteca chamada Highcharts. O Highcharts oferece uma biblioteca de tabelas desenvolvida em javascript, oferecendo tabelas e gráficos interativos [28]. Tanto o Bootstrap, quanto o Highcharts levam em consideração os navegadores mobile permitindo assim um redimensionamento de páginas e tabelas, um ambiente que permite o zoom e toques responsivos. O resultado pode ser observado na Figura 22:

Figura 22: Página Web com gráfico de temperatura interativo.

Nesta página, o usuário pode passar o mouse sobre os pontos para saber qual o valor de temperatura cada um representa, exportar o gráfico como imagem, além de possuir uma navegação fácil e amigável em ambientes mobiles.

4.1.4 OBSERVAÇÃO DE RECURSOS NO PROTOCOLO COAP

Uma das extensões do protocolo CoAP é a de “observar” recursos, isto é, buscar a representação de um recurso e mantê-la atualizada pelo servidor por um determinado período de tempo [29].

Utilizando um recurso periódico com o método GET, foi possível programar a função OBSERVE no servidor CoAP para buscar a temperatura do sensor do mote a cada 5 segundos. Caso essa temperatura fosse superior a 30 ºC, o status era alterado de OK para ALERT.

Essa implementação pode ser observada na Figura 23:

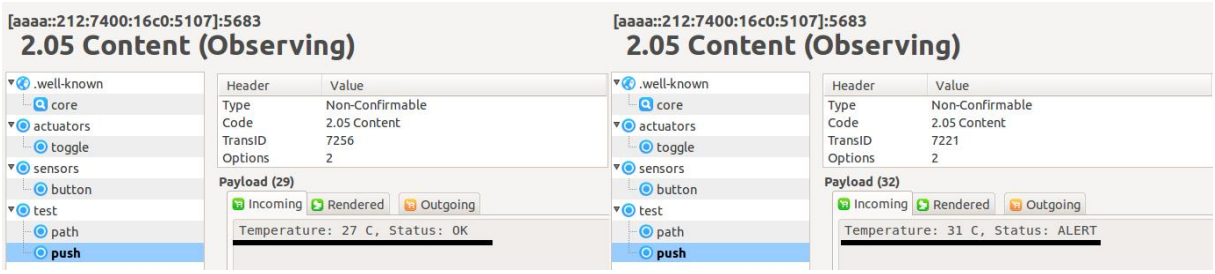

Figura 23: Função Observe do sensor de temperatura.

Do lado esquerdo da Figura 23, a temperatura medida pelo sensor foi 27 ºC, portanto, o status foi OK. Já, do lado direito, a temperatura foi 31 ºC, logo, o status foi alterado de OK para ALERT.

4.2 CAPTURA DE PACOTES COM O WIRESHARK

Para mostrar o funcionamento das aplicações e dos protocolos utilizados, foram realizadas algumas capturas com o auxílio do *Wireshark*.

O Wireshark é um software analisador de protocolos de rede que permite verificar o que está acontecendo em uma rede com grande detalhamento. O software suporta diversos tipos de protocolos, faz captura de pacotes em tempo real e análise offline, os dados de captura podem ser visualizados em uma interface gráfica e possui diversas opções extras com filtros de protocolos e exportação de capturas em vários formatos [30]. Uma imagem da interface gráfica do *Wireshark* pode ser observada abaixo:

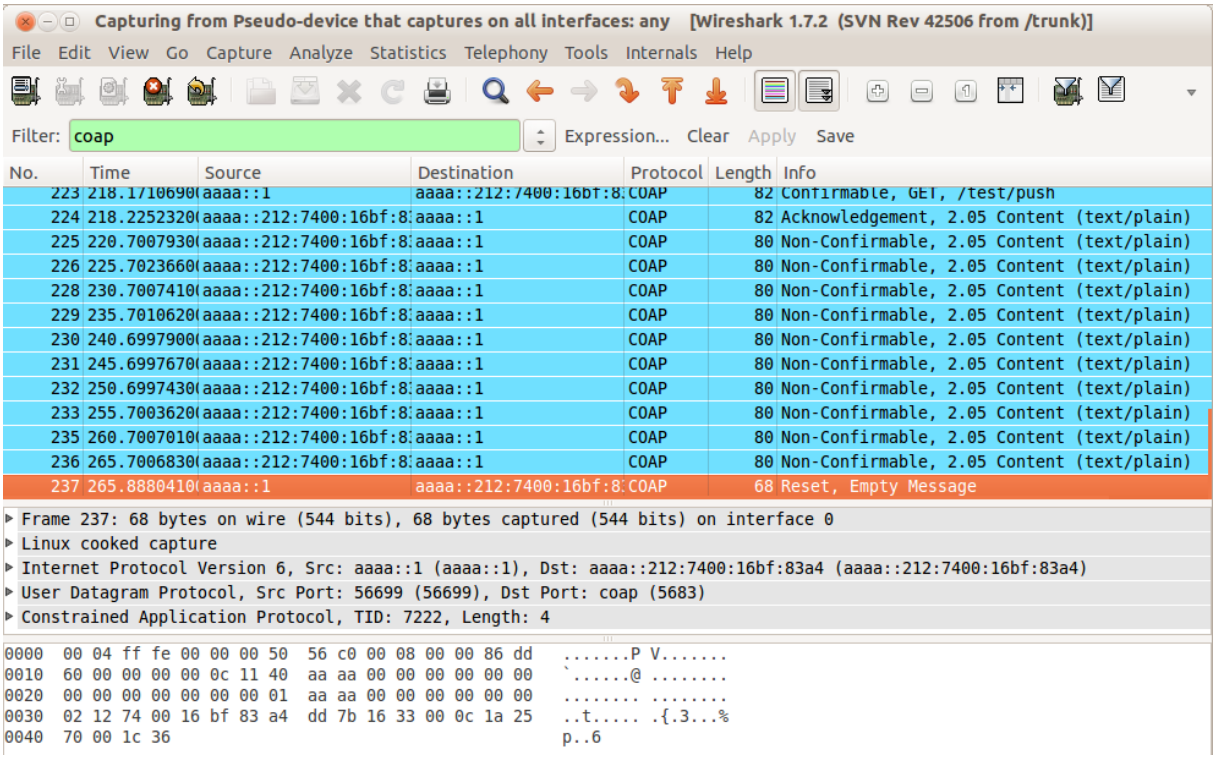

Figura 24: Interface gráfica do Wireshark durante a captura de pacotes.

Na Figura 24, encontra-se a interface de captura em tempo real de pacotes CoAP trocados entre o cliente e o servidor da aplicação desenvolvida. É possível ver o tempo de envio, a fonte, o destinatário, o tipo de protocolo, comprimento, informações e conteúdos.

Um pacote IPv6 obtido na captura demonstra que alguns cabeçalhos podem ser comprimidos descartando algumas informações redundantes, isso se confirma ao verificar os campos preenchidos com o valor 0. Também pode se observar que o

tipo de protocolo de transporte utilizado foi o UDP e que foi enviado um pacote CoAP de método GET confirmável. Os dados estão presentes na Tabela 3 seguinte:

Tabela 3: Informações de um pacote CoAP.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000	aaaa::1	aaaa::212:7400:16bf:83a4	COAP	85	Confirmable, GET, /.well-known/core
Internet Protocol Version 6, Src: aaaa::1 (aaaa::1), Dst: aaaa::212:7400:16bf:83a4 (aaaa::212:7400:16bf:83a4) 0110 .... = Version: 6 .... 0000 0000 .... = Traffic class: 0x00000000 .... 0000 00.. .... = Differentiated Services Codepoint: Default (0x00) ....0. .... = ECN-Capable Transport (ECT): Not set ....0 .... = ECN-CE: Not set .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000 Payload length: 29 Next header: UDP (17) Hop limit: 64 Source: aaaa::1 (aaaa::1) Destination: aaaa::212:7400:16bf:83a4 (aaaa::212:7400:16bf:83a4)						
Constrained Application Protocol, TID: 52297, Length: 21 01.. .... = Version: 1 ..00 .... = Type: Confirmable (0) .... 0010 = Option Count: 2 Code: GET (1) Transaction ID: 52297 Option #1: Uri-Path (Type: 9) 1001 .... = Delta: 9 .... 1011 = Length: 11 Uri-Path: .well-known Option #2: Uri-Path (Type: 9) 0000 .... = Delta: 0 .... 0100 = Length: 4 Uri-Path: core						

Algumas capturas de recursos utilizados nesse trabalho foram realizadas e analisadas como podem ser vistas na seção seguinte.

4.2.1 CAPTURA DO MÉTODO POST

Utilizando o recurso do atuador do LED da seção 4.1.1, pode-se capturar o método POST no Wireshark. A captura pode ser encontrada na próxima tabela.

Tabela 4: Captura do Método POST.

Time	Source	Destination	Protocol	Length	Info
712.548.812	aaaa::1	aaaa::212:7400:16bf:83a4	COAP	85	Confirmable, POST, /actuators/toggle
712.598.516	aaaa::212:7400:16bf:83a4	aaaa::1	COAP	68	Acknowledgement, 2.05 Content

Alguns pontos relevantes dessa captura são, além da presença do campo do método POST, pode-se ver que foi uma requisição confirmável e, assim que o pacote foi recebido com sucesso, uma notificação de recebimento foi enviada para a fonte (Acknowledgement).

4.2.2 CAPTURA DO MÉTODO GET

Com o recurso do sensor de bateria (descrito na seção 4.1.2), foi possível obter um valor estimado da carga da bateria. A captura desse recurso encontra-se na Tabela 5.

Tabela 5: Captura do Método GET.

Time	Source	Destination	Protocol	Length	Info
17.514.740.408	aaaa::1	aaaa::212:7400:16c0:5107	COAP	84	Confirmable, GET, /sensors/battery
Constrained Application Protocol, TID: 14811, Length: 20 01.. .... = Version: 1 ..00 .... = Type: Confirmable (0) .... 0010 = Option Count: 2 Code: GET (1) Transaction ID: 14811 Option #1: Uri-Path (Type: 9) 1001 .... = Delta: 9 .... 0111 = Length: 7 Uri-Path: sensors Option #2: Uri-Path (Type: 9) 0000 .... = Delta: 0 .... 0111 = Length: 7 Uri-Path: battery					
Time	Source	Destination	Protocol	Length	Info
17.514.799.769	aaaa::212:7400:16c0:5107	aaaa::1	COAP	85	Acknowledgement, 2.05 Content (text/plain)

```
Constrained Application Protocol, TID: 14811, Length: 21
  01.. .... = Version: 1
  ..10 .... = Type: Acknowledgement (2)
  .... 0001 = Option Count: 1
  Code: 2.05 Content (69)
  Transaction ID: 14811
  Option #1: Content-Type (Type: 1)
    0001 .... = Delta: 1
    .... 0000 = Length: 0
    [Expert Info (Warn/Malformed): Invalid Option Length: 0]
      [Message: Invalid Option Length: 0]
      [Severity level: Warn]
      [Group: Malformed]
  Payload Content-Type: text/plain (default), Length: 16, offset: 5
    Line-based text data: text/plain
      Battery 89 ( % )
```

Algumas informações interessantes obtidas nessa captura são que, assim como o POST observado anteriormente, a requisição foi confirmável e os campos do protocolo CoAP. Na requisição, está presente as opções selecionadas pelo usuário, do seguinte modo: Opção 1 sensors e Opção 2 battery. E, nas informações enviadas pelo servidor está presente um Payload contendo o texto: Battery 89 (%).

4.2.3 CAPTURA DA FUNÇÃO OBSERVE

A função OBSERVE foi implementada através de um recurso periódico e o método GET na seção 4.2.3.

Na Tabela 6, a função se inicia por um método GET em que o roteador se inscreve (SUBSCRIBE) para receber o recurso periodicamente. Logo em seguida, o servidor envia uma confirmação que o roteador irá receber esses recursos.

Na tabela 7, encontram-se dois pacotes recebidos dos recursos observados, essas informações são não-confirmáveis. No primeiro pacote, o conteúdo de texto do payload foi “Temperature: 28 C, Status: OK”, pois a temperatura estava abaixo de 30 ºC. No segundo, o payload continha “Temperature: 31 C, Status: ALERT”, já que a temperatura era superior a 30 ºC.

Na Tabela 8, é mostrado o fim da observação de recursos. Quando o usuário deseja encerrar a observação, ele pode clicar em Cancelar e o roteador envia ao servidor um RESET com uma mensagem vazia.

Tabela 6: Captura da Função OBSERVE (GET e ACKNOWLEDGMENT).

No.	Time	Source	Destination	Protocol	Length	Info
4227	8.584.261	aaaa::1	aaaa::212:7400:16c0:5107	COAP	82	Confirmable, GET, /test/push

Constrained Application Protocol, TID: 19120, Length: 18
- 01.. .... = Version: 1
- ..00 .... = Type: Confirmable (0)
- .... 0100 = Option Count: 4
- Code: GET (1)
- Transaction ID: 19120
- Option #1: Uri-Path (Type: 9)
  - 1001 .... = Delta: 9
  - .... 0100 = Length: 4
  - Uri-Path: test
- Option #2: Uri-Path (Type: 9)
  - 0000 .... = Delta: 0
  - .... 0100 = Length: 4
  - Uri-Path: push
- Option #3: Observe (Type: 10)
  - 0001 .... = Delta: 1
  - .... 0000 = Length: 0
  - [Expert Info (Warn/Malformed): Invalid Option Length: 0]
    - [Message: Invalid Option Length: 0]
    - [Severity level: Warn]
    - [Group: Malformed]
- Option #4: Token (Type: 11)
  - 0001 .... = Delta: 1
  - .... 0010 = Length: 2
  - Token: 432d

No.	Time	Source	Destination	Protocol	Length	Info
4228	8.584.557	aaaa::212:7400:16c0:5107	aaaa::1	COAP	82	Acknowledgement, 2.05 Content (text/plain)

Constrained Application Protocol, TID: 19120, Length: 18
- 01.. .... = Version: 1
- ..10 .... = Type: Acknowledgement (2)
- .... 0011 = Option Count: 3
- Code: 2.05 Content (69)
- Transaction ID: 19120
- Option #1: Content-Type (Type: 1)
  - 0001 .... = Delta: 1
  - .... 0000 = Length: 0
  - [Expert Info (Warn/Malformed): Invalid Option Length: 0]
    - [Message: Invalid Option Length: 0]
    - [Severity level: Warn]
    - [Group: Malformed]
- Option #2: Observe (Type: 10)
  - 1001 .... = Delta: 9
  - .... 0000 = Length: 0
  - [Expert Info (Warn/Malformed): Invalid Option Length: 0]
    - [Message: Invalid Option Length: 0]
    - [Severity level: Warn]
    - [Group: Malformed]
- Option #3: Token (Type: 11)
  - 0001 .... = Delta: 1
  - .... 0010 = Length: 2
  - Token: 432d
- Payload Content-Type: text/plain (default), Length: 9, offset: 9
  - Line-based text data: text/plain
    - Added 1/4

Tabela 7: Recebimento dos Recursos Durante a Observação.

No.	Time	Source	Destination	Protocol	Length	Info
4229	8.586.006	aaaa::212:7400:16c0:5107	aaaa::1	COAP	102	Non-Confirmable, 2.05 Content (text/plain)
Constrained Application Protocol, TID: 7213, Length: 38 01.. .... = Version: 1 ..01 .... = Type: Non-Confirmable (1) .... 0010 = Option Count: 2 Code: 2.05 Content (69) Transaction ID: 7213 Option #1: Observe (Type: 10) 1010 .... = Delta: 10 .... 0001 = Length: 1 Lifetime: 28 (s) Option #2: Token (Type: 11) 0001 .... = Delta: 1 .... 0010 = Length: 2 Token: 432d Payload Content-Type: text/plain (default), Length: 29, offset: 9 Line-based text data: text/plain Temperature: 28 C, Status: OK						
No.	Time	Source	Destination	Protocol	Length	Info
4262	8.616.006	aaaa::212:7400:16c0:5107	aaaa::1	COAP	105	Non-Confirmable, 2.05 Content (text/plain)
Constrained Application Protocol, TID: 7219, Length: 41 01.. .... = Version: 1 ..01 .... = Type: Non-Confirmable (1) .... 0010 = Option Count: 2 Code: 2.05 Content (69) Transaction ID: 7219 Option #1: Observe (Type: 10) 1010 .... = Delta: 10 .... 0001 = Length: 1 Lifetime: 31 (s) Option #2: Token (Type: 11) 0001 .... = Delta: 1 .... 0010 = Length: 2 Token: 432d Payload Content-Type: text/plain (default), Length: 32, offset: 9 Line-based text data: text/plain Temperature: 31 C, Status: ALERT						

Tabela 8: Fim da Observação de Recursos.

No.	Time	Source	Destination	Protocol	Length	Info
4285	8.706.006	aaaa::1	aaaa::212:7400:16c0:5107	COAP	68	Reset, Empty Message
Constrained Application Protocol, TID: 7237, Length: 4 01.. .... = Version: 1 ..11 .... = Type: Reset (3) .... 0000 = Option Count: 0 Code: Empty Message (0) Transaction ID: 7237						

5 CONCLUSÃO

Com o desenvolvimento desse trabalho foi possível estudar os mais diversos protocolos utilizados para a Internet das Coisas: IEEE 802.15.4, IETF 6LOWPAN, CoAP, assim como as aplicações no dia-a-dia e algumas implementações com o uso do hardware Tmote Sky e o sistema operacional Contiki.

Os protocolos possuem diversas particularidades e foram desenvolvidos pensando no uso eficiente de energia e tamanho de arquivos, pode-se notar que muitas aplicações possuem um modo de descanso e que os cabeçalhos nos pacotes são comprimidos, por exemplo.

O sistema operacional Contiki facilita muito o desenvolvimento de aplicações para IoT pois possui diversas ferramentas como o Cooja e exemplos prontos para teste. O Contiki não possui uma documentação muito bem elaborada e, em certos problemas, as soluções puderam ser encontradas em fóruns na internet.

Os protocolos estudados demonstraram-se compatíveis com as aplicações. Diversos hardwares disponíveis no mercado já suportam esses protocolos, facilitando a compatibilidade e desenvolvimentos futuros. Os protocolos e a arquitetura da Internet das Coisas também são compatíveis com os utilizados na internet atual, permitindo a comunicação entre eles.

A partir da implementação de aplicações da Internet das Coisas, pode-se validar a utilidade do seu estudo e desenvolvimento. Também foi possível desenvolver aplicações que podem ser utilizadas no dia-a-dia para solucionar e auxiliar a vida das pessoas de maneira eficiente e simples.

Com a captura de pacotes utilizando o Wireshark, foi possível verificar como ocorre o transporte de pacotes entre dois hardwares reais na Internet das Coisas e as mensagens que são trocadas durante a comunicação.

Embora não tenha sido abordado nesse trabalho, o fator segurança nos softwares e hardwares na Internet das Coisas é de extrema relevância e importância e seu estudo e avaliação poderiam ser realizados em uma continuação desse projeto.

Outra possível melhoria no projeto seria a comunicação dos hardwares de Internet das Coisas com a Internet de fato. Enviando informações em tempo real dos hardwares para a rede, sendo possível requisitá-las através de um website, por exemplo.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[1] ITU. ICT Facts and Figures: The world in 2015. Disponível em: <http://www.itu.int/en/ITU-D/Statistics/Pages/facts/default.aspx>. Acesso em: 10 abr. 2016.

[2] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, Imrich Chlamtac, Internet of things: Vision, applications and research challenges, Ad Hoc Networks, Volume 10, Issue 7, September 2012, Pages 1497-1516, ISSN 1570-8705, http://dx.doi.org/10.1016/j.adhoc.2012.02.016.
(http://www.sciencedirect.com/science/article/pii/S1570870512000674)

[3] Luigi Atzori, Antonio Iera, Giacomo Morabito, The Internet of Things: A survey, Computer Networks, Volume 54, Issue 15, 28 October 2010, Pages 2787-2805, ISSN 1389-1286, http://dx.doi.org/10.1016/j.comnet.2010.05.010.
(http://www.sciencedirect.com/science/article/pii/S1389128610001568)

[4] U.S. National Intelligence Council (NIC), Global Trends 2025: A Transformed World, NIC, Available online: www.dni.gov/nic/NI-2025-project.html, November 2008.

[5] M. R. Palattella *et al*., "Standardized Protocol Stack for the Internet of (Important) Things," in *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1389-1406, Third Quarter 2013.
doi: 10.1109/SURV.2012.111412.00158

[6] A. Zanella, N. Bui, A. Castellani, L. Vangelista and M. Zorzi, "Internet of Things for Smart Cities," in *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22-32, Feb. 2014.

doi: 10.1109/JIOT.2014.2306328

[7] COMPUTERWORLDUK. Boeing 787s to create half a terabyte of data per flight, says Virgin Atlantic. Disponível em: <http://www.computerworlduk.com/news/data/boeing-787s-create-half-terabyte-of-data-per-flight-says-virgin-atlantic-3433595/>. Acesso em: 10 abr. 2016.

[8] COMPUTERWORLDUK. Internet of things examples: 12 best uses of IoT in the enterprise. Disponível em: <http://www.computerworlduk.com/galleries/cloud-computing/internet-of-things-best-business-enterprise-offerings-3626973/#6>. Acesso em: 15 abr. 2016.

[9] NEST. Meet the Nest Thermostat. Disponível em: <https://nest.com/thermostat/meet-nest-thermostat/>. Acesso em: 18 abr. 2016.

[10] AMAZON. Amazon Dash Button. Disponível em: <http://www.amazon.com/b?node=10667898011>. Acesso em: 18 abr. 2016.

[11] MICRIUM. IoT Protocol Stack Options: People Internet vs. Device Internet. Disponível em: <https://www.micrium.com/iot/internet-protocols/>. Acesso em: 20 abr. 2016.

[12] Geoff Mulligan. 2007. The 6LoWPAN architecture. In *Proceedings of the 4th workshop on Embedded networked sensors* (EmNets '07). ACM, New York, NY, USA, 78-82. DOI=http://dx.doi.org/10.1145/1278972.1278992

[13] HEATH JR, Robert W.. Multi-Hop Networking. Disponível em: <http://www.profheath.org/research/multi-hop-networking/>. Acesso em: 21 abr. 2016.

[14] INTELLIGENCE, Security. The Importance of IPv6 and the Internet of Things. Disponível em:

<https://securityintelligence.com/the-importance-of-ipv6-and-the-internet-of-things/>. Acesso em: 21 abr. 2016.

[15] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, Transmission of IPv6 Packets over IEEE 802.15.4 Networks, RFC 4944, Internet Engineering Task Force RFC 4944, September 2007.

[16] J. Hui and P. Thubert, Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks, RFC 6282, Internet Engineering Task Force RFC 6282, September 2011.

[17] Olsson, J., 6LowPan Desmystified, Texas Instruments White Paper (2014).

[18] Tsvetkov, T., RPL: IPv6 Routing Protocol for Low Power and Lossy Networks, Seminar SN SS (2011).

[19] INTERNET ENGINEERING TASK FORCE. The Constrained Application Protocol (CoAP). Disponível em: <https://tools.ietf.org/html/rfc7252>. Acesso em: 01 maio 2016.

[20] CONTIKI OS. Contiki: The Open Source OS for the Internet of Things. Disponível em: <http://www.contiki-os.org/>. Acesso em: 03 maio 2016.

[21] CONTIKI OS. Get Started with Contiki. Disponível em: <http://www.contiki-os.org/start.html>. Acesso em: 13 jul. 2016.

[22] ADVANTICSYS. MTM-CM5000-MSP. Disponível em: <http://www.advanticsys.com/shop/mtmcm5000msp-p-14.html>. Acesso em: 30 jul. 2016.

[23] ADVANTICSYS. CM5000 DATASHEET, 2010. Disponível em: <http://www.epssilon.cl/files/EPS5000.pdf>. Acesso em: 31 jul. 2016.

[24] KOVATSCH, Matthias. Copper (Cu): CoAP user-agent for Firefox. Disponível em: <http://people.inf.ethz.ch/mkovatsc/copper.php>. Acesso em: 10 jul. 2016.

[25] SEHGAL, Anuj. Using the Contiki Cooja Simulator., Jacobs University Bremen, Bremen, 2013. Disponível em: <http://cnds.eecs.jacobs-university.de/courses/iotlab-2013/cooja.pdf>. Acesso em: 10 jul. 2016.

[26] FIELDING, Roy Thomas. Representational State Transfer (REST). Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>. Acesso em: 14 jul. 2016.

[27] IBM. Entenda o Representational State Transfer (REST) no Ruby. Disponível em: <http://www.ibm.com/developerworks/br/library/os-understand-rest-ruby/>. Acesso em: 20 jul. 2016.

[28] HIGHSOFT. Highcharts: About Us. Disponível em: <http://www.highcharts.com/about>. Acesso em: 21 jul. 2016.

[29] INTERNET ENGINEERING TASK FORCE. Observing Resources in the Constrained Application Protocol (CoAP). Disponível em: <https://tools.ietf.org/html/rfc7641>. Acesso em: 13 fev. 2017.

[30] WIRESHARK. Wireshark. 2017. Disponível em: <https://www.wireshark.org/>. Acesso em: 20 fev. 2017.