UNIVERSIDADE FEDERAL DE SANTA CATARINA

Gerenciamento autonômico de dispositivo de Internet of Things com restrição de energia utilizando arquitetura de Fog Computing

Ana Luiza Córdova de Jesus

Florianópolis - SC

2018/2

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

Gerenciamento autonômico de dispositivo de Internet of Things com restrição de energia utilizando arquitetura de Fog Computing

Ana Luiza Córdova de Jesus

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Florianópolis - SC

2018/2

Gerenciamento autonômico de dispositivo de Internet of Things com restrição de energia utilizando arquitetura de Fog Computing

Ana Luiza Córdova de Jesus

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Dr. Carlos Becker Westphall

Co-orientador: Me. Hugo Vaz Sampaio

Banca Examinadora:

_____

Dra. Carla Merkle Westphall

_____

Me. Hugo Vaz Sampaio

_____

Bel. Leandro Loffi

Resumo:

Com o crescimento da *Internet of things* nos últimos anos, houve também o crescimento da *Fog computing*. Uma das maneiras de utilizar as duas tecnologias de modo a beneficiar os usuários é aplica-las em casas inteligentes, de modo a tornar o uso de dispositivos comuns de forma automatizada, e muitas vezes trazendo uma economia de energia. Com isso em mente, é proposto a criação de um protótipo de alarme de incêndio inteligente, utilizando conceitos de *Internet of things* e *Fog* computing, de modo a medir o tempo de vida útil desse protótipo, de acordo com o gasto de energia do mesmo. Finalmente, são realizadas comparações de cálculos de tempo de vida, variando o protocolo de rede utilizado (ZigBee, Wi-Fi e Bluetooth).

Palavras Chaves: *Fog Computing*, *Internet of things*, economia de energia, protocolos de rede.

SUMÁRIO

LISTA DE QUADROS

LISTA DE FIGURAS

LISTA DE SIGLAS

IoT – *Internet of Things* (Internet das Coisas)

6LoWPAN – *IPv6 over Low-Power Wireless Personal Area Networks*

6GLAD – *Global to Link-layer Address Translation for 6LoWPAN Overhead Reduction*

AT – Modo de funcionamento transparente da antena XBee ZigBee

API – Modo de funcionamento *Application programming* interface da antena XBee ZigBee

RX – Porta Serial de Recebimento

TX – Porta Serial de Transmissão

PAN ID - *Personal Area Network ID*

GND – Ligação *ground* no Arduino

ASCII – *American Standard Code for Information Interchange*

DHT11 – Sensor de umidade e temperatura

MQ-2 – Sensor de gás e fumaça

LPG – Gás liquefeito de petróleo

UFSC – Universidade Federal de Santa Catarina

LRG – Laboratório de Redes e Gerência

Mbps – Megabytes por segundo

1. INTRODUÇÃO

Na seção são apresentados os conceitos iniciais do trabalho, tal como a motivação, justificativa e objetivos, tanto gerais do trabalho quanto mais específicos.

1.1 Motivação

É inegável o crescimento do uso de dispositivos inteligentes na sociedade atual. Uma das formas de aplicar essa tecnologia e fazer uso de dispositivos inteligentes de maneira a garantir a segurança de seus usuários, como por exemplo, utilizar conceitos e tecnologia de casas inteligentes e *Internet of Things* para situações de emergência, como por exemplo, um incêndio. Um alarme de incêndio inteligente conectado à rede de uma residência, que utilize formas de captura e envio de dados para um usuário e que ainda assim tenha uma preocupação com o gasto de energia desse dispositivo é algo que une a crescente onda de *Internet of Things* e *Fog Computing*.

1.2 Justificativa

O uso de tecnologias de *Internet of Things* pode ser uma forma de garantir segurança de seus usuários. Uma das formas de realizar isso é focar na prevenção de incêndios; tendo em vista a quantidade de acidentes que ocorrem e que se transformam em tragédias - como foi o caso do incêndio que destruiu o Museu Nacional no Rio de Janeiro (Abreu et al, 2018) - sem que haja nenhum sistema de monitoramento e/ou aviso no caso de uma situação de emergência. Com isso, foi pensado em uma solução para evitar esse tipo de acidente, utilizando *Internet of Things* e *Fog Computing* de modo a ser possível realizar o monitoramento e envio de dados para o usuário em tempo real.

1.3 Objetivos

A seguir é apresentada a descrição dos objetivos do trabalho como um todo, contendo ambos objetivos gerais e específicos.

1.3.1 Objetivo Geral

Desenvolver um sistema em que seja possível capturar dados referentes a uma situação de incêndio e realizar o envio desses dados capturados para o usuário, utilizando um ambiente de *Fog Computing*.

1.3.2 Objetivos Específicos

- Apresentar um protótipo de alarme de incêndio inteligente.
- Estimar o tempo médio de vida do nó IoT restrito de energia usando baterias de diferentes potências.
- Comparar as estimativas de tempo de vida de baterias de acordo com o protocolo de rede utilizado.

1.4 Organização do trabalho

O trabalho encontra-se organizado da seguinte maneira – no capítulo 2 são apresentados os conceitos fundamentais utilizados ao longo do trabalho (*Fog Computing* e *Internet of Things*). No capítulo 3 é realizada a revisão bibliográfica sistemática do trabalho, explicando como foram obtidas as referências utilizadas no embasamento teórico do trabalho, além de discorrer sobre alguns dos problemas referentes ao uso de *Fog Computing* e *Internet of Things* (principalmente quando relacionados ao consumo de energia) e de apresentar o conceito de *Smart Homes*,

que une ambas tecnologias. No capítulo 4 é apresentado o desenvolvimento de um protótipo de alarme de incêndio, baseado nos conceitos apresentados, envolvendo o projeto de hardware, o projeto de rádio e o protocolo criado para troca de mensagens. No capítulo 5 é apresentado o protocolo de envio de pacotes entre o nó *Mist*, a *Fog* e o nó IoT. No capítulo 6 são realizados cálculos de estimativas de tempo de vida do nó IoT, baseando-se nos gastos dos componentes utilizados no nó, e é feita uma comparação desse tempo quando utilizado diferentes protocolos de rede (ZigBee, Wi-Fi e Bluetooth). Finalmente, o capítulo 7 apresenta as conclusões e possibilidades de trabalhos futuros.

2. CONCEITOS FUNDAMENTAIS

Abaixo são apresentados os dois principais conceitos utilizados no trabalho - *Fog Computing* e *Internet of Things*, assim como suas principais características e relevância ao tema de economia de energia.

2.1 *Fog Computing*

Fog Computing (ou Computação na Neblina, em português), pode ser considerada como uma extensão da computação em nuvem, cujo foco é em aplicações que precisam de poucos atrasos, e análises em tempo real (Singh, 2016). A definição mais formal define *Fog Computing* como “um modelo em camadas que permite o acesso onipresente a recursos computacionais escaláveis” (Iorga et al, 2018). Este modelo é composto de nodos que podem ser físicos ou virtuais, e que se encontram localizados entre dispositivos considerados inteligentes e serviços centralizados de computação na nuvem (Iorga et al, 2018).

Seu nome vem de uma das principais diferenças com a computação em nuvem - a idéia que a neblina se encontra mais próxima do chão do que a nuvem, e isso se reflete em uma maior proximidade do usuário com o servidor (Botta et al, 2016).

Até 2020 estima-se que entre 20 e 50 bilhões de dispositivos novos serão conectados à Internet, gerando um aumento de 3 trilhões de dólares à economia mundial (Varghese et al, 2017). Dessa forma, a *Fog Computing* começa a se estabelecer como uma ótima alternativa à computação na nuvem, principalmente no que se diz em respeito a *Internet of Things*.

Porém, necessita-se afirmar que *Fog Computing* não vem para substituir completamente a computação em nuvem - ao contrário, a utilização de *Fog* vem para suprir algumas questões de difícil resolução na nuvem, assim como utilizar suas

características únicas. Algumas destas características são em relação a segurança (utilizando uma distância menor para envio de dados entre dispositivos, tem-se uma chance mais reduzida desses dados serem interceptados), e eficiência (a computação na neblina faz uso de diversos tipos de dispositivos, podendo utilizar os mesmo até seu maior potencial de computação e armazenamento) (Chiang et al, 2017). Uma característica em especial que merece um destaque maior é a cognição, ou seja, a habilidade da arquitetura de *Fog* em reconhecer e escolher onde melhor realizar funções de computação e de armazenamento - seja isso no *Edge*, nos próprios dispositivos ou caso seja necessário, na nuvem. Isso se deve ao fato de aplicações que são construídas utilizando *Fog Computing* estão mais próximas ao usuário e podem então determinar mais facilmente como e onde realizar as operações necessárias (Chiang et al, 2017).

Uma questão importante quando é utilizado *Fog*, principalmente ao que pertence ao uso de *Internet of Things*, é a heterogeneidade dos dispositivos utilizados. *Routers*, *switches*, *access points*, e mesmo *smartphones* podem ser utilizados no ambiente de *fog* ao invés de tecnologias mais tradicionais como *data centers*. Isso traz desafios específicos que devem ser analisados com maior cuidado, como a questão de segurança e escalabilidade tanto vertical quanto horizontal (Varghese et al, 2017).

2.2 Internet of Things

Ainda que se tenha certa dificuldade em conseguir uma definição acadêmica formal e universal, devido ao fato que a *Internet of Things* (IoT, Internet das Coisas em português) é considerada como um paradigma relativamente novo na área de telecomunicações modernas (Atzori; Iera; Morabito, 2010). A ideia básica, porém, pode ser definida como uma força omnipresente onde objetos e tecnologias como

sensores, celulares, etiquetas com identificação por radiofrequência, entre outros, interagem e trabalham entre seus vizinhos com a finalidade de atingir um objetivo comum (Atzori; Iera; Morabito, 2010).

Algumas aplicações em IoT utilizam direção assistida (carros inteligentes), mapas online com realidade aumentada, casas inteligentes (como possibilidade de monitorar o interruptor de lâmpadas, sistema de aquecimento, sistemas de alarme, etc), uso em redes sociais para atualização automática, e diversas outras (Atzori; Iera; Morabito, 2010).

Uma arquitetura básica composta de *Fog Computing* e dispositivos de *Internet of Things* pode ser exemplificada pela Figura 1.

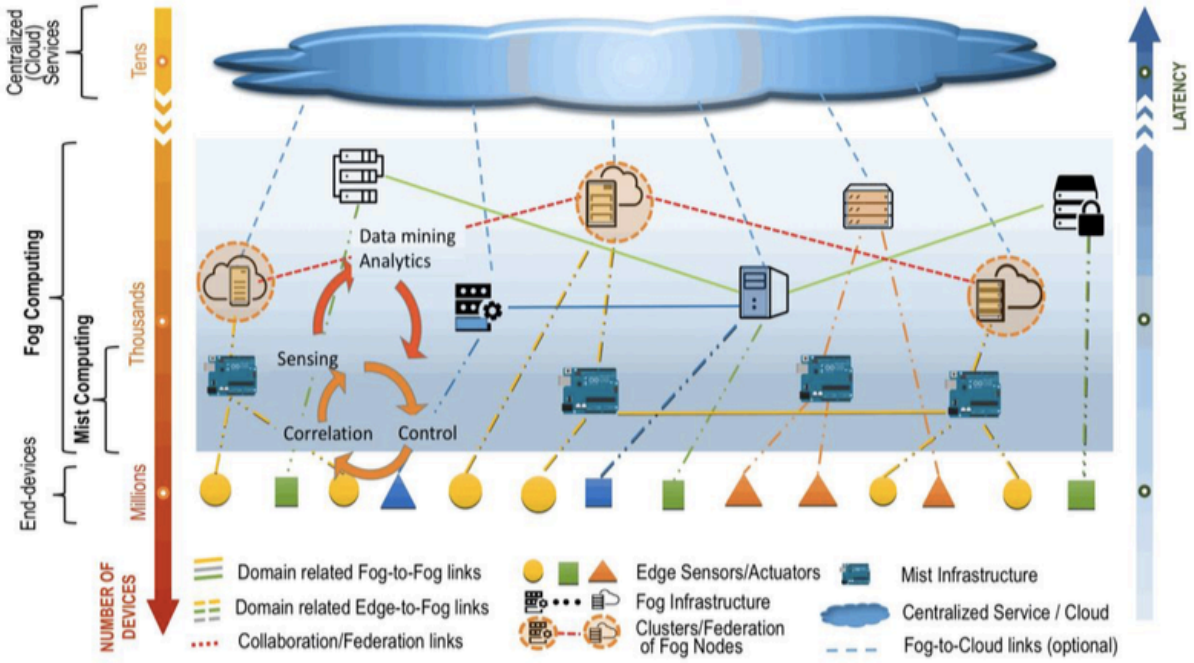


Figura 1 – Arquitetura unindo *Fog computing* e *Internet of things* (Iorga et al, 2018).

Big Data é uma preocupação relacionada a *Internet of Things*, devido ao grande volume de dados que são gerados através de dispositivos conectados a rede de IoT (Botta et al, 2016). Atualmente tem-se cerca de 8,4 bilhões de dispositivos conectados à Internet, enviando e recebendo dados, sendo que esse número pode

deve se encontrar na faixa dos 20 bilhões em 2020 (Marques, 2017). Somente em 2017 mais de 1 bilhão de dispositivos novos foram conectados à Internet pela primeira vez (IEA 4E, 2018). Esses dispositivos são de diversos tipos e formatos, desde *smart TVs* e smartphones, até automóveis e casas inteligentes (Marques, 2017). O potencial financeiro da difusão da *Internet of Things* também é algo que deve ser considerado, pois estima-se que até o ano 2025 a mesma deve ter um impacto de $11 trilhões ao ano - o equivalente a 11% da economia mundial (Manyika et al, 2015). As áreas da economia mais afetadas, e consequentemente, mais beneficiadas com o uso amplo e difundido da IoT são fábricas industriais, cidades inteligentes, monitoramento de doenças em humanos, direção autônoma de veículos, entre outros (Manyika et al, 2015).

No Brasil, é possível enxergar um movimento crescente quanto ao interesse de empresas que querem desenvolver soluções com o uso da *Internet of Things* - em São Paulo existem mais de 20 projetos em andamento de startups e pequenas empresas para os mais diversos fins, que foram financiados por um programa da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) (Marques, 2017).

Com esse crescimento da IoT, é preciso também preocupar-se com os impactos ambientais - a área de Tecnologia da Informação e Comunicação sozinha contribui com 2% do aumento do efeito estufa. No Brasil, porém, ainda são poucas as empresas que se preocupam com o TI verde (De Faria; Siqueira; Da Silva Martins, 2016). Embora tenha-se muitas organizações que se preocupem e tomem medidas relacionadas a diminuição de gases de efeito estufa, poucas tratam sobre a questão de energia elétrica utilizada no uso de servidores e data centers em seus materiais destinados ao publico em geral (De Faria; Siqueira; Da Silva Martins, 2016).

3. REVISÃO BIBLIOGRÁFICA SISTEMÁTICA

Foi realizada uma revisão bibliográfica sistemática, de modo a adotar um método científico para a escolha e consequente resumo dos estudos selecionados para apresentação do estado da arte.

3.1 Objetivo da revisão

O objetivo da revisão foi procurar por estudos sobre *Fog Computing* e *Internet of Things*, dando uma preferência maior para trabalhos que abordassem a integração das duas tecnologias. Além disso, houve também a necessidade de buscar pesquisas científicas que se preocupassem com a questão de economia de energia e apresentasse o estado da arte deste assunto.

3.2 Identificação da Literatura

Foi decidido utilizar trabalhos tanto em inglês quanto em português, porém devido ao volume de publicações no antecedente, foi dada prioridade a trabalhos em inglês. Esses então, foram recuperados através do banco de dados Google Scholar.

Inicialmente, foram utilizados apenas termos altamente abrangentes como "*Fog Computing*" e "*Internet of Things*", que trouxeram 105.000 e 369.000 resultados, respectivamente. A partir desses resultados originais, foram feitas outras pesquisas com diversas expressões, principalmente quando relacionadas a protocolos de rede, para refinar a busca.

O Quadro 1 apresenta os resultados numéricos de pesquisas científicas através de buscas realizadas no Google Scholar, dependendo dos termos utilizados. Decidiu-se por limitar o período de publicação dos trabalhos entre 2010 e 2018.

Dentro desses resultados foram escolhidos os trabalhos que seriam utilizados para a elaboração desta pesquisa.

Palavra chave	Total de Resultados	Categoria
Fog Computing	105.000	Categoria 1
Internet of Things	369.000	Categoria 1
Network Protocols	1.820.000	Categoria 1
Internet of Things, Network Protocols	39.600	Categoria 2
Fog Computing, Network Protocols	17.000	Categoria 2
Fog Computing, Internet of Things	15.500	Categoria 3
Fog Computing, Smart Home	10.500	Categoria 3
Internet of Things, Smart Home	191.000	Categoria 3
Fog Computing, Internet of Things, Network Protocols	11.400	Categoria 4
Fog Computing, Internet of Things, Network Protocols, Smart Home	12.500	Categoria 5

Quadro 1 – Resultados de pesquisas no banco Google Scholar

Os resultados obtidos mostram que, em sua maioria, quando são realizadas novas buscas com mais palavras chave, há uma diminuição no número de resultados obtidos, e, portanto, uma diminuição no escopo de trabalho. Porém, como exibido na categoria 5, há um aumento no número de resultados, embora tenha-se adicionado mais palavras chave no intuito de encontrar pesquisas mais especificas e demonstrar a importância e utilidade deste trabalho. Com isso, é possível que seja feita uma união de resultados com as palavras chaves, ao invés de uma intersecção como seria esperado.

3.3 Seleção dos estudos a serem incluídos

A identificação de estudos foi realizada através de seu título e resumo. Para efetivamente decidir pela inclusão do mesmo, foram determinados parâmetros que os trabalhos deveriam aderir:

- Estar escrito em inglês ou em português
- Período de publicação entre 2010 e 2018.
- Texto integral disponível para download, dando preferência a arquivos em formato PDF.
- Uso de palavras chaves no titulo, resumo ou mesmo desenvolvimento do estudo.
- Contribuição relevante ao desenvolvimento deste trabalho.

Foram também determinados parâmetros para exclusão de estudos:

- Não disponibilização do texto integral para download.
- Estudo sem contribuição para o contexto deste trabalho, mesmo que apresentasse palavras chaves em seu titulo, resumo ou mesmo desenvolvimento.

3.4 Análise e discussão

Os estudos encontrados a partir dos critérios acima ainda sofreram uma outra análise para que fossem selecionados como fontes bibliográficas neste trabalho, dependendo do conteúdo apresentado nos mesmos e a vertente que este pretende seguir. Por fim, foram selecionados 18 estudos, que formam a base referencial deste trabalho (cujas referências completas se encontram na seção de Referências Bibliográficas) e em que foi possível notar três aspectos importantes – consumo e economia de energia, uso de protocolos de rede e automação residencial.

3.4.1 Consumo e economia de energia

Economia de energia de dispositivos de IoT, principalmente quando integrando *Fog Computing*, pode ser visto de diversos ângulos. Primeiramente, deve-se ter em mente que nem todas aplicações tiram vantagem do uso de *Fog* - aplicações e serviços que necessitem de muitos *downloads*, *updates* ou necessitem de computações que são consideradas muito pesadas irão economizar mais energia quando enviadas à *Cloud*, sem tentar realizar essas tarefas na borda da rede antes (Jalali et al, 2017). Dessa forma, é melhor focar em aplicações de tempo real e baixa latência quando for decidido tirar vantagem das características diferenciadas do *Fog*.

Uma técnica comum utilizada em *Cloud Computing*, consumo de energia ocioso, não é tão facilmente atingida com *Fog*, devido a existência de menos recursos como CPU, armazenamento e RAM na *Fog* quando comparado a *Cloud*. Assim, é mais difícil utilizar as mesmas técnicas para economia de energia, fazendo com que, caso servidores de *Fog* se tornem inativos por longos períodos de tempo, *Fog* não consiga ser tão eficiente na economia de energia quando comparando com a *Cloud* (Jalali et al, 2017). Dessa forma, é preciso pensar em outras formas possíveis de economizar energia quando integrando *Fog* e *Internet of Things.*

Outra questão importante quando se fala de economia de energia utilizando Internet das Coisas, é o fato que muitos dispositivos de IoT dependem de baterias para funcionar e também se encontram conectados a redes de comunicação sem fio. Para usuários desses dispositivos, uma longa vida útil de um dispositivo é considerada importante; ou seja, o dispositivo dependente da bateria deve conseguir durar muitos anos (Friedli et al, 2016). O uso de Wi-Fi, por exemplo, é um protocolo de uso comum em dispositivos de IoT; seu uso permite a conexão de um dispositivo

à Internet estimado em um ano, utilizando duas pilhas alcalinas do tipo AA, quando utilizando gerenciamento eficiente de energia (Perera et al, 2017).

3.4.2 Uso de protocolos de rede para economia de energia

Protocolos de rede, embora não sejam pensados como a principal fonte de gasto de energia em dispositivos de *Internet of things*, podem desempenhar um papel importante na identificação de fontes de consumo de energia em um dispositivo, assim como na ineficiência da energia do mesmo (Biason, 2017).

O uso de protocolos de rede como Bluetooth e ZigBee podem ser boas opções para economia de energia em *Fog Computing* (Jalali et al, 2017). Deve-se ter em mente, porém, que diferentes dispositivos de IoT podem utilizar diferentes tecnologias a seu favor para uma maior economia de energia. Por exemplo, ZigBee pode ser utilizado em sensores, atuadores e mesmo lâmpadas LED smart (embora necessite de um *gateway*), mas não é considerado apropriado para uso em gateways e câmeras de monitoramento (Friedli et al, 2016).

Mesmo o uso de protocolos de roteamento como IPv4 ou IPv6 podem apresentar um gasto de energia quando conectando dispositivos de IoT com a Internet (Zimmerman et al, 2008). Dessa forma, é preciso usar protocolos como o 6LoWPAN (*IPv6 over Low-Power Wireless Personal Area Networks*) ou então 6GLAD (*Global to Link-layer Address Translation for 6LoWPAN Overhead Reduction),* onde também é possível realizar a compressão de cabeçalhos de endereçamento que, dependendo de parâmetros tais como tamanho do pacote, topologia de rede, escopo de comunicação, entre outros, podem diminuir até 38% do consumo de energia nas baterias de dispositivos inteligentes (Zimmerman et al, 2008).

3.4.3 Automação Residencial

O uso de automação residencial (ou casas inteligentes) pode ser visto como uma das áreas em que a Internet das Coisas pode ser aplicada, visto que o uso de dispositivos inteligentes do cotidiano - tal como lâmpadas, portas e janelas - pode ser caracterizado como utilização de dispositivos de Internet das Coisas (Stojkoska; Trivodaliev, 2017).

Atualmente, existe um terreno muito fértil quando se fala de automação residencial. Isso significa tornar o uso de objetos comuns do cotidiano mais inteligente, com o uso de tecnologias de comunicação, para uma melhora na qualidade de vida dos habitantes de uma residência. Ou seja, automação residencial utiliza tecnologias como conexão com rede (usando Bluetooth, Wi-Fi, e outros protocolos de rede) para conectar dispositivos de uso doméstico tal como luzes, portas, cortinas, sistema de ventilação, entre outros, e realizar o uso dos mesmos de forma mais inteligente. O exemplo mais simples e um dos mais utilizados é o uso de lâmpadas inteligentes que podem ser ligadas, desligadas e ter sua luminosidade controlada através do uso de aplicativos em dispositivos móveis como o telefone celular (Ghazal et al, 2015).

O uso da automação residencial pode ser facilmente integrado com *Fog Computing*, devido a integração de *Fog* com IoT já discutido; no caso da automação residencial, os mesmos paradigmas, vantagens e desafios ainda se aplicam.

Com o crescimento de dispositivos IoT, e consequentemente a existência de mais dispositivos domésticos que podem ser considerados inteligentes, é necessário pensar em casos de economia de energia com esse tipo de dispositivo (Stojkoska; Trivodaliev, 2017).

Stojkoska e Trivodaliev (2017) mencionam que embora a integração de tecnologias de IoT com o conceito de *smart homes* (e no caso, envio para *Cloud computing* para processamento com maior poder do que é possível na *Fog*) pode trazer muitos benefícios, o gerenciamento desse sistema é melhor feito de modo a realizar cada tarefa em cada um dos níveis citados – de modo a ter um maior proveito de cada um deles. A Figura 2 mostra essa situação.

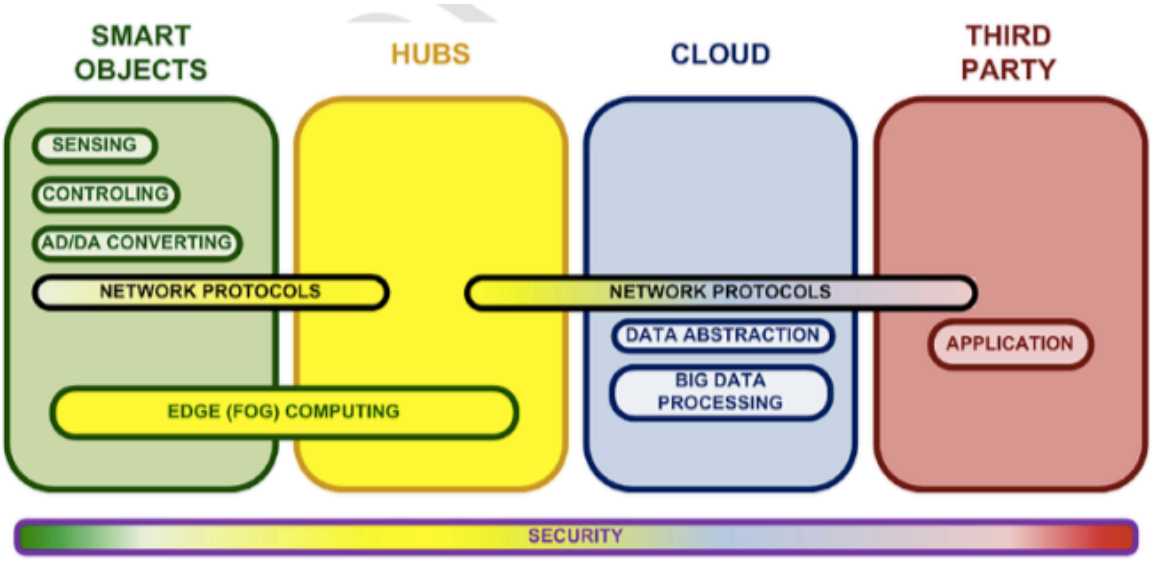


Figura 2 – Gerenciamento de sistema de smart home (Stojkoska; Trivodaliev, 2017).

Nesse caso, seria importante trabalhar com cada tarefa a nível de dispositivo de *smart home*; *hubs* (que no caso da integração com *Fog Computing* pode ser considerado o ambiente de *Fog*); a *Cloud* para armazenamento e processamento de dados; e finalmente, aplicações para funcionamento e regulação da operação dos dispositivos propriamente ditos (regular a operação de lâmpadas inteligentes, configurar o funcionamento de sistemas de ar-condicionado, entre outros). No caso de terceiras partes, a diferença de tarefas realizada por terceiras partes e pelos dispositivos inteligentes propriamente ditos tem a ver com a possibilidade de uso de algoritmos mais complexos para realizar a mineração e extração de conhecimento armazenado na *Cloud* para a criação de recomendações e perfis personalizados para

cada tipo de usuário na residência que utiliza esse tipo de tecnologia (Stojkoska; Trivodaliev, 2017).

4. DESENVOLVIMENTO DO SISTEMA

A partir desse capítulo é delineado o sistema desenvolvido neste trabalho, que inclui o projeto de *hardware* do protótipo de alarme de incêndio, programa de *software* para a captura de dados e protocolo de envio desses dados capturados para a Fog.

Tendo em mente a necessidade da longevidade de baterias em dispositivos IoT, assim como diminuir o consumo de energia dos mesmos, é proposto o desenvolvimento de um protótipo de dispositivo IoT protótipo com utilização de placa microcontroladora, antena e sensores, de forma a considerar as limitações e consumo energético do hardware. Fazendo uso da *Internet of Things* e do conceito de automação residencial, o protótipo é de um alarme de incêndio inteligente para detecção de valores de umidade e temperatura, gás e fumaça e presença de chamas, cujos dados capturados são enviados para um nó coordenador na *Fog* para pré-processamento dos dados.

Com isso, é possível realizar uma análise do gasto de energia do protótipo, estimando o tempo de vida do nó quando utilizando baterias para funcionamento do mesmo.

Por fim, são realizadas comparações de gasto de energia quando utilizando diferentes protocolos de rede - ZigBee, Wi-Fi e Bluetooth.

A arquitetura utilizada no desenvolvimento do projeto é exibida na Figura 3.

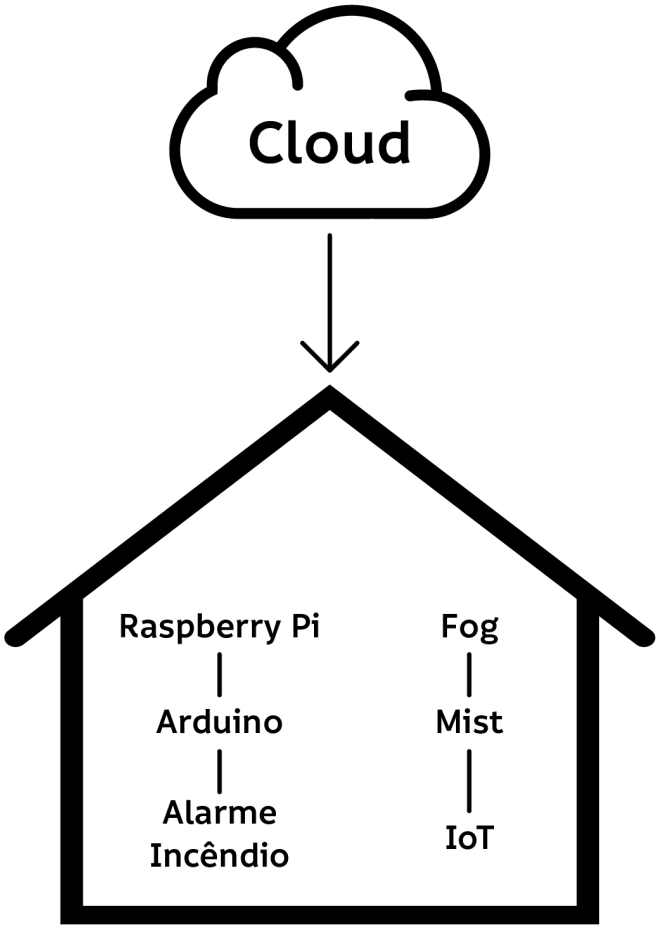


Figura 3 – Arquitetura utilizada no trabalho.

4.1 Projeto de nós

O projeto dos nós é dividido em duas partes, sendo que a primeira diz respeito a parte de hardware do alarme de incêndio, e a segunda parte é relativa a rádio, de modo a transmitir os dados capturados pela parte de hardware em uma rede sem fio.

Para o desenvolvimento dos nós do protótipo foi escolhido utilizar peças com alta disponibilidade no mercado, e que também possuem preços acessíveis. Dessa forma, a placa de prototipação selecionada foi o Arduino Uno com microcontrolador ATmega328P (Arduino, 2018). A antena escolhida para uso no protótipo foi a Xbee ZigBee, que é fabricada de acordo com as especificações do protocolo ZigBee (Digi,

2018). Todos os nós da rede, sendo eles nós sensores, coordenador e roteador, utilizaram uma placa Arduino e uma antena Xbee ZigBee.

4.1.1 Projeto da parte de hardware

O Arduino Uno (Arduino, 2018) foi selecionado como *hardware* principal do protótipo. O Arduino Uno é uma placa de microcontrolador baseada no ATmega328P, possuindo 14 portas digitais de entrada e saída, 6 portas de entrada analógica, botão de reset da placa, conexão via USB, e freqüência de clock de 16 MHz.

Três sensores foram utilizados no protótipo, um sensor de gás e fumaça (MQ-2), um sensor de chama, e um sensor de temperatura e umidade do ar (DHT11).

O dispositivo IoT também possui a capacidade de ser atuador; assim, caso sejam detectados níveis/valores acima de um limite pré-estabelecido, serão acionados avisos sonoros (com o uso de um buzzer), um aviso visual (com o uso de um LED), e será enviada uma mensagem para a *Fog* indicando perigo.

4.1.2 Projeto da parte de rádio

O protocolo ZigBee é um protocolo baseado no padrão IEEE 802.15.4, de baixa velocidade de transmissão e baixo consumo de energia, principalmente quando comparado com outros protocolos amplamente utilizados, como Wi-Fi e Bluetooth (Chen; Azhari; Leu, 2018). Dessa forma o mesmo foi escolhido para ser utilizado no desenvolvimento do protótipo IoT.

Em redes ZigBee, diferentes dispositivos podem cumprir diferentes papéis, dependendo do seu tipo (ZigBee Alliance, 2014). Esses três diferentes papéis são:

1) Coordenador ZigBee: inicia a formação da rede e escolhe o canal da rede, PAN ID (*Personal Area Network ID*) e PAN ID extendido para identificação da rede. Dessa

forma, é possível utilizar várias redes em um mesmo local físico (Sampaio e Motoyama, 2017).

2) Roteador ZigBee: os roteadores são chamados para difundir o tráfego dos outros dispositivos da rede. Como os dispositivos finais podem entrar em estado de hibernação, os roteadores ZigBee (assim como os coordenadores Zigbee), podem armazenar em *cache* informações de descoberta em nome de seus nós filhos de dispositivo final quando os mesmos se encontram em suspensão.

3) Dispositivos Finais ZigBee (End Devices ZigBee): dispositivos finais são aqueles que se conectam a rede de modo a enviar e receber pacotes de dados para nós pais. É possível que dispositivos finais entrem em modo de hibernação para economia de energia. Dispositivos finais podem ser sensores para capturar dados como temperatura, umidade do ar, entre outros.

Para o componente de rádio propriamente dito, foi escolhido utilizar a antena Xbee ZigBee, especialmente devido a compatibilidade de uso com o protocolo ZigBee, além do protocolo Bluetooth. A antena Xbee Zigbee é fabricada pela empresa Digi, de acordo com as especificações do protocolo ZigBee. O componente pode ter até 14 nós filhos, com alcance máximo de 60 metros em ambientes urbanos (fechados). O Xbee ZigBee possui também freqüência de 2.4 GHz, com velocidade de transmissão de 250 Kbps (Digi, 2018).

O uso do Xbee também permite o download e uso do software XCTU desenvolvido pela Digi, para a configuração da antena e outros nós na rede. Com o uso do XCTU é possível estabelecer qual o modo de funcionamento de cada antena (Coordenador ZigBee, Roteador Zigbee ou Dispositivo Final), assim como o PAN ID, o endereço de destino de envio dos pacotes e mesmo realizar a montagem de pacotes (Digi, 2018).

A configuração do nó coordenador do protótipo IoT, relacionando-o com o nó sensor, no software XCTU é exibida na Figura 4.

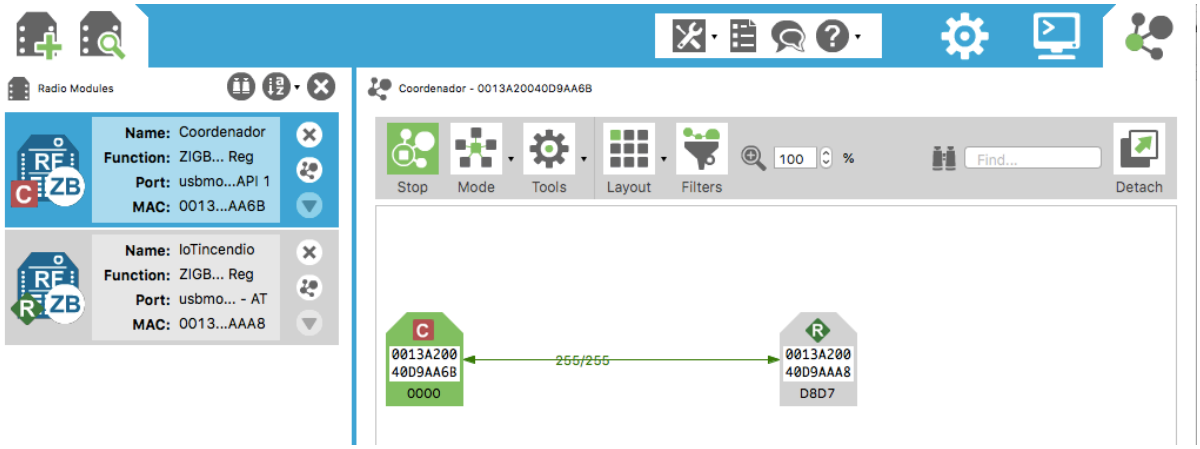


Figura 4 – Configuração dos nós no software XCTU.

4.2 Projeto de Nó Coordenador e Nó Roteador

Os nós que servem como coordenador e roteador no protótipo possuem apenas ligações entre a antena Xbee e a placa Arduino Uno. Essas ligações são realizadas entre as portas de transmissão e recebimento no Arduino e na antena (TX e RX, respectivamente, em ambos os componentes), assim como ligações de energia para o funcionamento dos componentes - conexão de 3.3 volts que é necessária para ligar a antena Xbee, assim como ligação de ground (GND) entre a antena e o Arduino. Essas conexões físicas são exemplificadas na Figura 5.

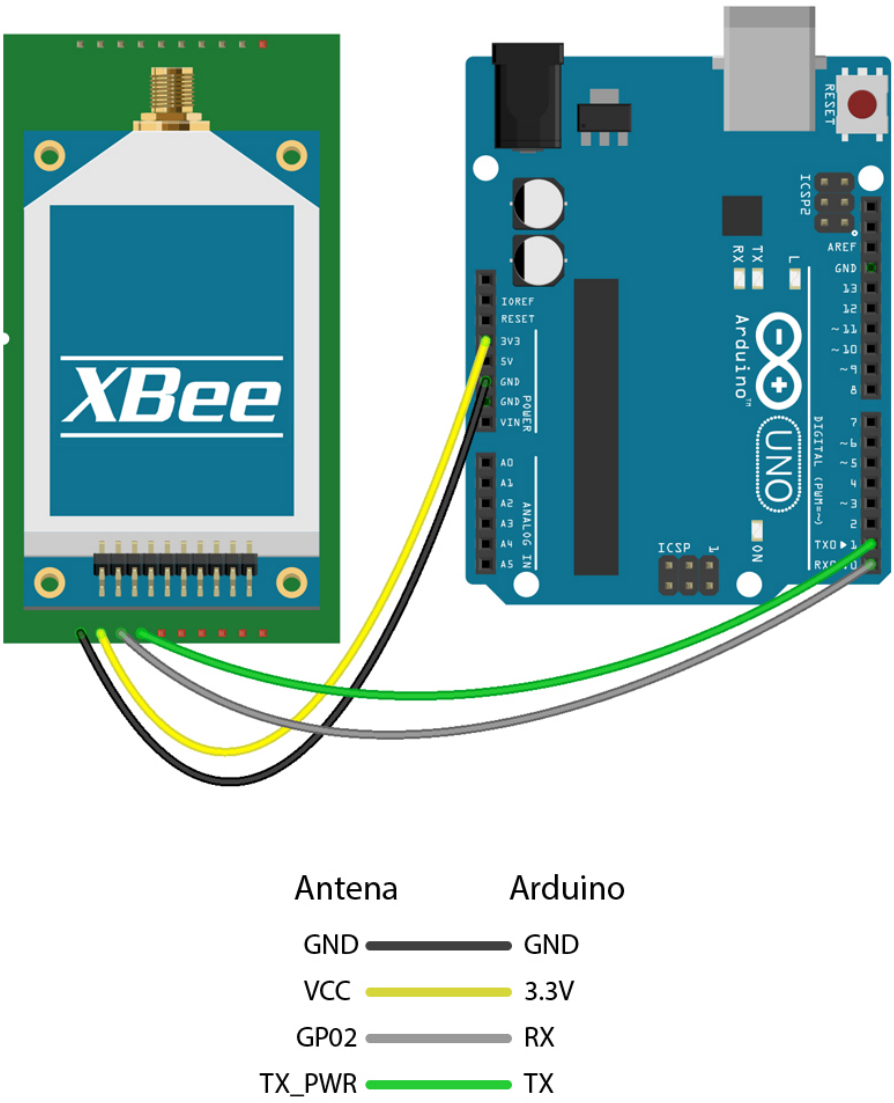


Figura 5 – Conexão física entre o Arduino UNO e a antena XBee ZigBee

Existe também a necessidade de conexão entre as portas seriais da antena e do Arduino na mesma velocidade, pelo fato que as portas dos componentes são seriais. A porta serial da antena XBee pode ser configurada com taxas de 115200 bits/seg, portanto a porta do Arduino UNO deve ser configurada a uma velocidade de 115200 bits/seg. (Sampaio e Motoyama, 2017).

A antena XBee pode funcionar em dois modos diferentes, AT e API, configurados pelo software XCTU:

1) **Modo AT (Transparente):** usado quando for necessário o envio de dados de um nó sensor para o nó coordenador. Ao receber esses dados através da porta serial RX, a antena configurada como nó coordenador encapsula os dados de forma automática, enviando-os para o endereço de destino configurado no software XCTU. Exemplificando, é possível enviar o conteúdo “x” para a porta serial RX da antena XBee com o comando “Serial.print(x);”, quando utilizando o Arduino UNO. Dessa forma, esse conteúdo “x” é encapsulado automaticamente e enviado para o endereço de destino (Sampaio e Motoyama, 2017).
2) **Modo API (*Application programming interface*):** usado para envio de dados para diferentes endereços de destino. Com o uso do modo API, é necessária a montagem manual de pacotes de envio, byte a byte. Por exemplo, o byte inicial de um pacote de transmissão usando o protocolo ZigBee é o valor hexadecimal 0x7E, e para transmissão do mesmo para a porta RX da antena, é necessário utilizar o comando “Serial.write(0x7E);” (Sampaio e Motoyama, 2017).

4.3 Montagem de Pacotes

Em nós sensores com a antena XBee no modo AT, o envio de dados capturados para o coordenador é realizado tal que, quando um nó sensor captura um valor, esse é armazenado na variável “x” no microcontrolador da placa do Arduino. Depois disso, o microcontrolador executa o comando “Serial.print(x);”, de forma a enviar a variável “x” da porta TX do Arduino para a porta RX da antena. A antena recebe esse valor, como um sinal imprimível, conforme a tabela ASCII (*American Standard Code for Information Interchange*), e procede a encapsular

automaticamente o valor capturado pelo sensor e então envia-lo para o nó coordenador (Sampaio e Motoyama, 2017).

Os pacotes enviados de um nó sensor para o coordenador possuem tamanhos variáveis de acordo com a quantidade de dados que devem ser enviados; esse tipo de pacote é chamado de “pacote de requisição de transmissão”.

Para exemplificar o modo de transmissão de dados, pode-se pensar em uma situação hipotética onde, em um nó sensor, foram capturados os valores 25 graus Celsius e 74% de umidade no sensor DHT11, valor 99% de presença de gás e/ou fumaça no sensor MQ-2 e o valor 1 (indicando fogo) no sensor de chama. Esses valores serão armazenados em variáveis separadas “temperatura”, “umidade”, “gasTotal” e “chama”, identificados pelos comandos do Quadro 2, onde é possível enviar para a porta TX do Arduino os valores capturados, que então serão enviados a porta RX da antena, onde será realizada a leitura dos dados.

```
void enviar () {
   Serial.print(umidade);
   Serial.print(temperatura);
   Serial.print(sensorReading);
   Serial.print(valorSensorFumaca);

 }
```

Quadro 2 – Comandos para envio de dados capturados pelo nó sensor

A antena por sua vez, transformará os valores decimais capturados em valores hexadecimais, um caractere por vez, separadamente. Ou seja, o valor decimal 25 será transformado para os valores hexadecimais 0x32 e 0x35, onde cada caractere ocupa 1 byte de tamanho no pacote de transmissão. O mesmo acontece com o restante dos valores.

Após a transformação dos valores capturados, eles serão encapsulados de maneira automática em um pacote, e finalmente enviados para o nó coordenador. A Figura 6 mostra um exemplo de um pacote criado com o comando enviarPacote(), com tamanho de 25 bytes.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
7E	00	15	10	01	00	13	A2	00	40	D9	AA	6B	FF	FE	00	00	32	35	37	34	39	39	31	99

Figura 6 – Exemplificação de um pacote de requisição de transmissão.

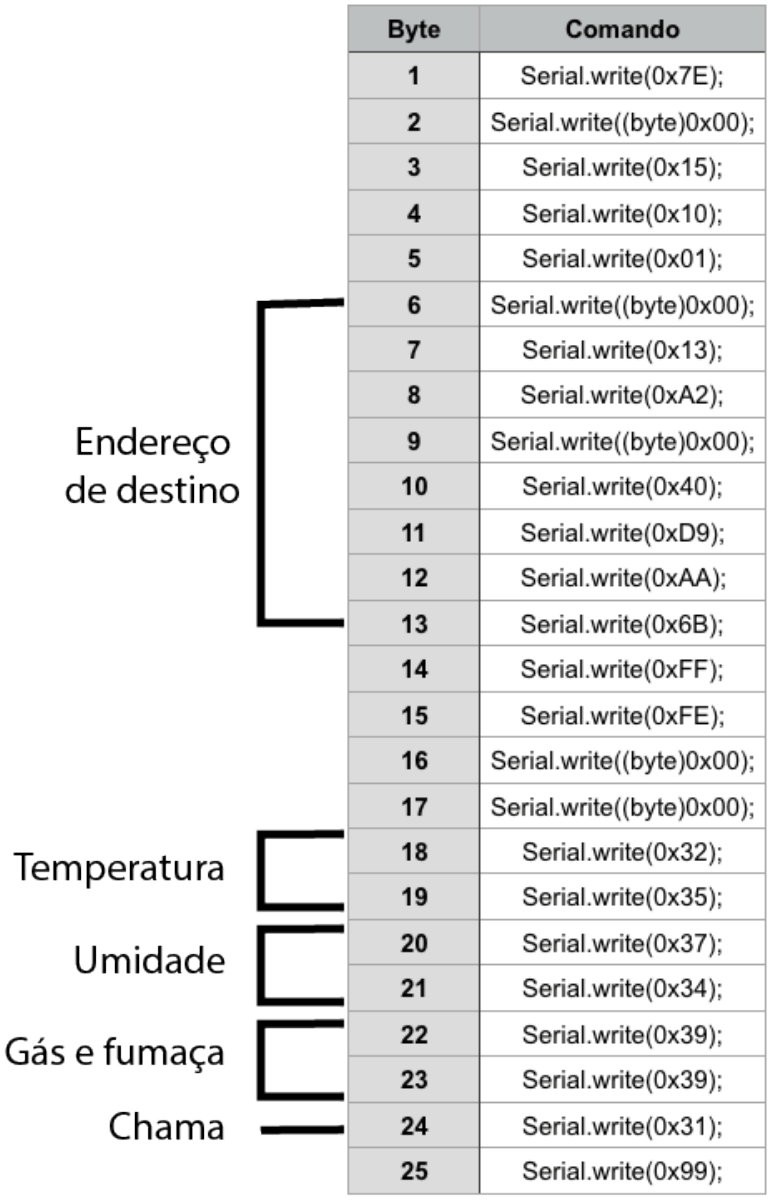
O Quadro 3 mostra com detalhes o pacote da Figura 6. Deve-se ressaltar que os campos de ID do frame, endereços de 64 e 16 bits, raio de *broadcast* e opções foram definidos previamente durante a configuração da antena com o XCTU.

Byte	Descrição	Valor Hexadecimal	Significado
1	Delimitador inicial	7E	
2 e 3	Tamanho do pacote	00 15	21 bytes de tamanho
4	Tipo do frame	10	Requisição de transmissão
5	ID do frame	01	Identificação do frame
6 a 13	Endereço de destino de 64-bit	00 13 A2 00 40 D9 AA 6B	Endereço do nó coordenador
14 e 15	Endereço de destino de 16-bit	FF FE	Envio para todos os roteadores
16	Raio de broadcast	00	Pulos máximos de broadcast
17	Opções	00	
18 a 24	Dados	32 35 37 34 39 39 31	Dados capturados pelo nó sensor, com valores da tabela ASCII
25	Checksum	19	

Quadro 3 – Detalhes de um pacote de requisição de transmissão

Para os nós sensores que utilizam a antena XBee no modo API, é necessário que seja realizada a montagem dos pacotes para o nó coordenador de forma manual; o encapsulamento dos dados enviados não é realizado de maneira automática como acontece com a utilização do modo AT da antena. A vantagem disso é a possibilidade de modificar o pacote de forma direta no momento de sua criação (Sampaio e Motoyama, 2017).

Quando o nó sensor precisar realizar o envio de pacotes para o nó coordenador, é necessário escrever o pacote byte a byte, seguindo os padrões do tipo de pacote. Por exemplo, todos os pacotes de requisição iniciam sempre com o valor hexadecimal 0x7E, dessa forma é necessário utilizar um comando “Serial.write(0x7E);” para que a antena identifique o primeiro byte do pacote. O restante do pacote também deve ser enviado dessa mesma maneira; um exemplo disso se encontra na Figura 7, enviando os 18 bytes comuns do pacote, além de mais 7 bytes referentes aos dados enviados pelo nó sensor (2 bytes referentes ao valor de temperatura, 2 bytes referentes ao valor de umidade, 2 bytes referentes ao valor de gás e fumaça, e 1 bytes referentes a presença ou não de chamas), totalizando 25 bytes enviados.



Byte	Comando
1	Serial.write(0x7E);
2	Serial.write((byte)0x00);
3	Serial.write(0x15);
4	Serial.write(0x10);
5	Serial.write(0x01);
6	Serial.write((byte)0x00);
7	Serial.write(0x13);
8	Serial.write(0xA2);
9	Serial.write((byte)0x00);
10	Serial.write(0x40);
11	Serial.write(0xD9);
12	Serial.write(0xAA);
13	Serial.write(0x6B);
14	Serial.write(0xFF);
15	Serial.write(0xFE);
16	Serial.write((byte)0x00);
17	Serial.write((byte)0x00);
18	Serial.write(0x32);
19	Serial.write(0x35);
20	Serial.write(0x37);
21	Serial.write(0x34);
22	Serial.write(0x39);
23	Serial.write(0x39);
24	Serial.write(0x31);
25	Serial.write(0x99);

Figura 7 – Comandos necessários para criação de pacote de requisição de transmissão ZigBee.

Quando o pacote é recebido no nó coordenador, a antena processa o pacote e na porta RX é feita a substituição dos bytes de endereço de destino por bytes indicando o nó que enviou o pacote, e envia em sua porta RX para porta TX do Arduino (Sampaio e Motoyama, 2017).

O envio dos dados entre o nó sensor e o nó coordenador pode ser exemplificado pela Figura 8, onde é exibido o envio de um pacote de transmissão de

dados capturados pelo nó sensor. Quando o nó coordenador recebe esse pacote, ele realiza o processamento do mesmo, assim como envio para a porta serial. A porta serial deve processor e extrair as informações capturadas pelo sensor, podendo estar conectada a um computador ou a uma placa Arduino (Sampaio e Motoyama, 2017).

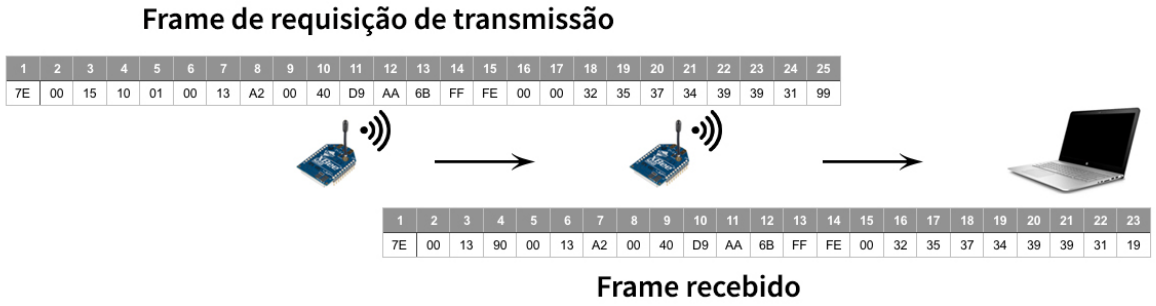


Figura 8 – Exemplificação de envio de dados do nó sensor para o nó coordenador

É necessário notar que, ao ser recebido pelo nó coordenador, o pacote “perde” dois bytes - referentes aos campos Raio de Broadcast e ID do frame – pois os mesmos são utilizados apenas por pacotes de envio de transmissão, não em pacotes recebidos. O Quadro 5 mostra os detalhes de um pacote de recebimento.

Byte	Descrição	Valor Hexadecimal	Significado
1	Delimitador inicial	7E	
2 e 3	Tamanho do pacote	00 13	19 bytes de tamanho
4	Tipo do frame	90	Frame recebido
5 a 12	Endereço de origem de 64-bit	00 13 A2 00 40 D9 AA 6B	Endereço do nó sensor
13 e 14	Endereço de origem de 16-bit	FF FE	Endereço local do nó sensor
15	Opções	00	
16 a 22	Dados	32 35 37 34 39 39 31	Dados capturados pelo nó sensor
23	Checksum	19	

Quadro 5 - Detalhes de um pacote de recebimento.

A montagem manual dos pacotes pode ser realizada de modo simples, através do uso de uma ferramenta (XBee Frame Generator) presente dentro do software XCTU, que auxilia na geração de pacotes de diversos tipos. Assim, a criação e

montagem do pacote pode ser realizada de modo mais fácil e eficiente, também evitando erros. Um exemplo do uso da ferramenta é mostrado na Figura 9, com a montagem de um pacote de transmissão, como explicado anteriormente.

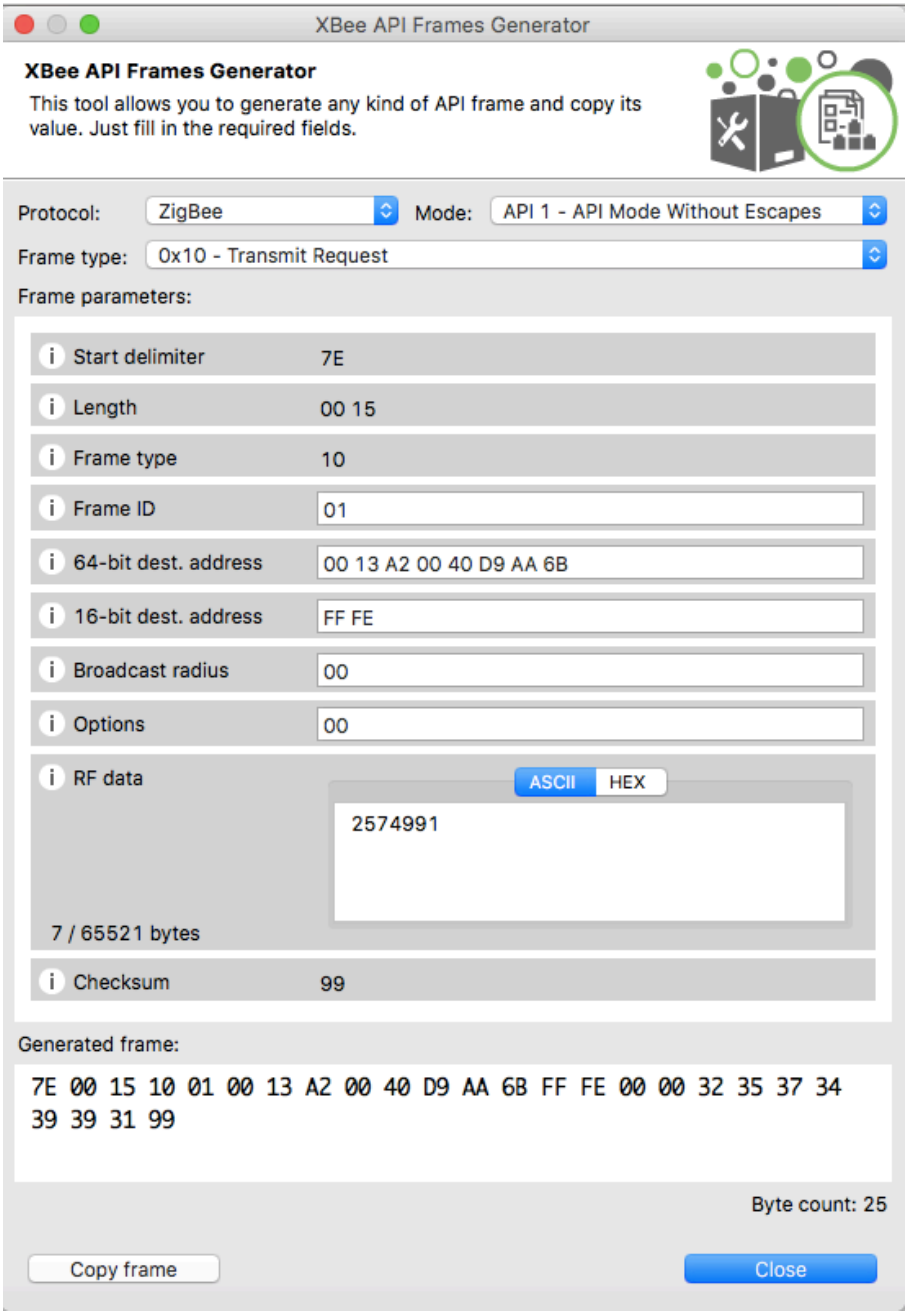


Figura 9 – Montagem de pacote de transmissão no XCTU.

Para este trabalho, o nó sensor irá utilizar a antena XBee em modo AT, enquanto que o nó coordenador irá utilizar a antena XBee em modo API.

4.4 Nós Sensores

Foram selecionados três sensores para uso no protótipo: de temperatura e umidade, de chamas, e de fumaça e gases.

A medição de temperatura e umidade no nó sensor foi realizada através do uso do sensor DHT11. O DHT11 possui componentes que conseguem detectar umidade e temperatura; quanto a umidade, o DHT11 possui uma acurácia de ± 5% de umidade relativa e quanto a temperatura, o sensor possui uma precisão de ± 2°C (Aosong, 2018). Esse sensor foi conectado ao Arduino através de uma porta analógica, e a leitura dos dados é feita de maneira simples devido ao uso de uma biblioteca para Arduino, o que permite a leitura dos valores de umidade em porcentagem e os valores de temperatura em graus Celsius. Isso encontra-se exemplificado no Quadro 4, com o método capturarTemperaturaUmidade().

```
void capturarTemperaturaUmidade () {
        digitalWrite(7, HIGH);

        umidade = dht.readHumidity();
        temperatura = dht.readTemperature();

        digitalWrite(7, LOW);

        Serial.print("Umidade: ");
        Serial.print(umidade);
        Serial.print("Temperatura: ");
        Serial.print(temperatura);

}
```

Quadro 4 – Método para capturar dados de detecção de umidade e temperatura

A conexão entre o DHT11, a placa Arduino e a antena ZigBee é exemplificada na Figura 10.

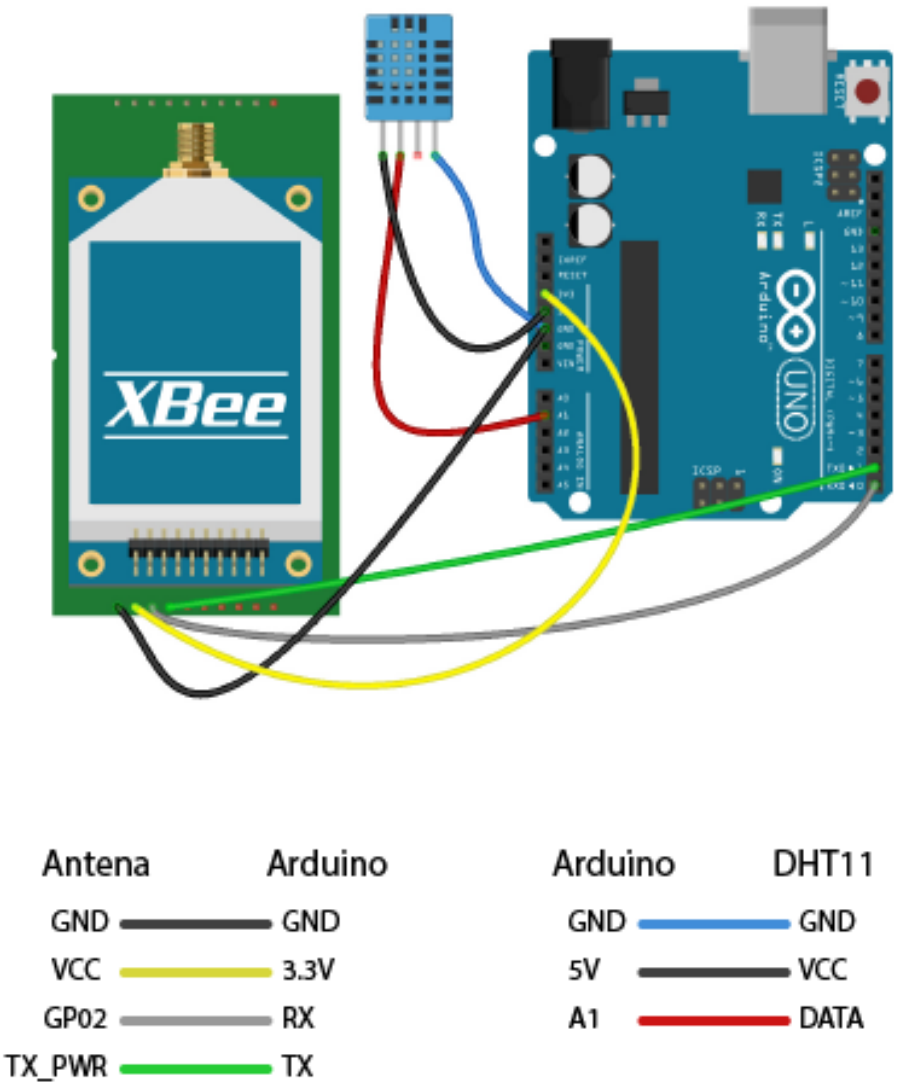


Figura 10 – Conexão física do nó sensor de umidade e temperatura

O nó sensor de fumaça e gases possui um sensor do tipo MQ-2, capaz de detectar os gases LPG (Gás liquefeito de petróleo), butano, propano, metano, álcool, hidrogênio, além de conseguir detectar fumaça em um ambiente (Hanwei, 2018). Foi escolhido utilizar o MQ-2 ao invés de outros sensores da família MQ pois ele se mostrou mais completo para a situação de um protótipo de alarme de incêndio inteligente, no quesito de gases capturados, além da possibilidade de detectar a presença de fumaça.

As valores capturados nas portas analógicas do Arduino são representados por 10 bits, assim os valores estarão entre 0 e 1023; é possível realizar um mapeamento no código do Arduino para que os valores sejam mapeados entre 0 e 100% de gás e/ou fumaça detectados no ar. Isto se encontra exemplificado no Quadro 5, com o método capturarGasFumaca().

```
Void capturarGasFumaca() {
        digitalWrite(11, HIGH);

        valorSensorFumaca = analogRead(pinSensorGas);
        valorSensorFumaca = map(valorSensorFumaca, 0, 1023, 0, 100);

        digitalWrite(11, LOW);

}
```

Quadro 5 – Método para capturar dados de detecção de gás e fumaça.

Foram realizados experimentos em um ambiente controlado, utilizando um isqueiro comum. Com o pressionamento da válvula de gás do isqueiro, foram detectados valores de concentração de gás cerca de 30% pelo sensor.

Uma representação da conexão física desse nó sensor se encontra na Figura 11.

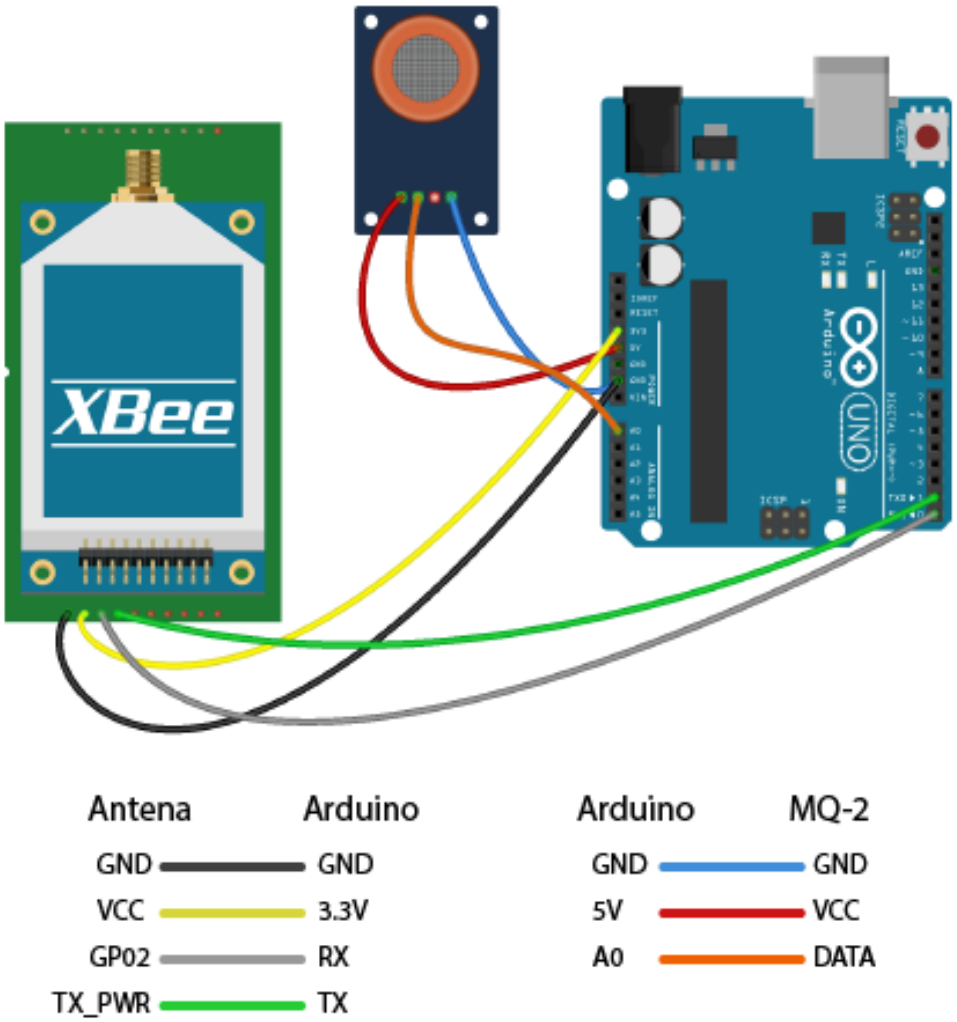


Figura 11 – Conexão física do nó sensor de fumaça e gases.

Para detecção de fogo propriamente dito, foi utilizado um sensor de chamas. Devido ao uso do componente de epóxi preto no sensor, o mesmo tem capacidade de detecção de chamas e outras fontes de calor, cujo tamanho de onda seja entre 760 a 1100 nm (Filipe Flop, 2018). A captura de dados desse sensor se encontra exemplificado no Quadro 6, com o método capturarChamas().

```
void capturarChamas() {
        digitalWrite(10, HIGH);

        sensorReading = analogRead(pinSensorChama);

        if (sensorReading < 1000) {
                sensorReading = 1;
                } else {
        sensorReading = 0;
}

        Serial.print(sensorReading);
digitalWrite(10, LOW);

}
```

Quadro 6 – Método para captura de dados de detecção de chamas.

Foram realizados testes em ambiente controlado, com utilização de um isqueiro comum, para emissão de chamas. O sensor conseguiu capturar a uma distância aproximada de 30 centímetros da fonte de calor. Para uma situação de emergência, seria necessário ambiente com segurança apropriada para esse tipo de teste. Uma representação da conexão física desse nó sensor se encontra na Figura 12.

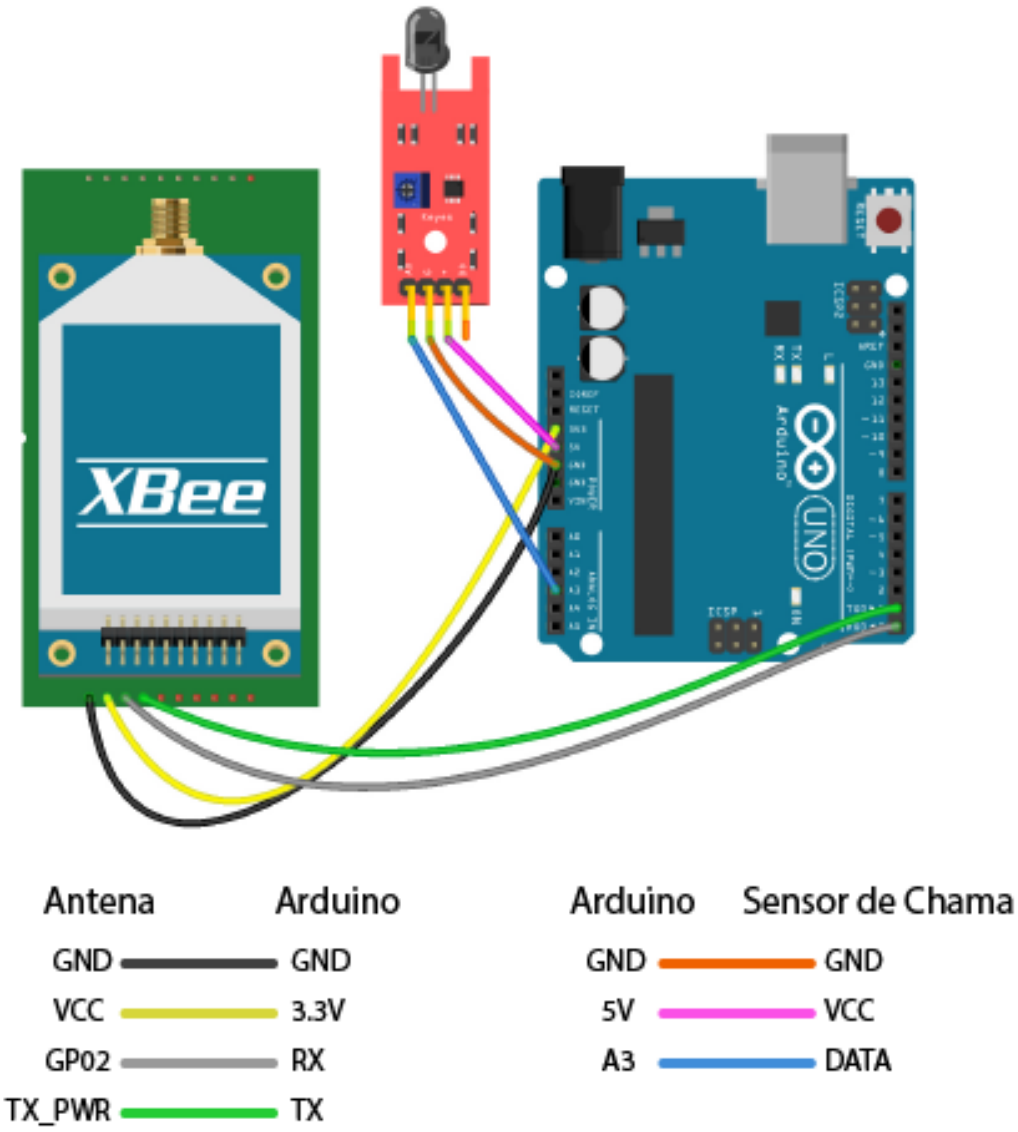


Figura 12 – Conexão física do sensor de chamas.

Por fim, foram utilizados alertas físicos no evento de disparo do alarme, na forma de um LED e um buzzer passivo. Uma representação da conexão física desses elementos se encontra na Figura 13.

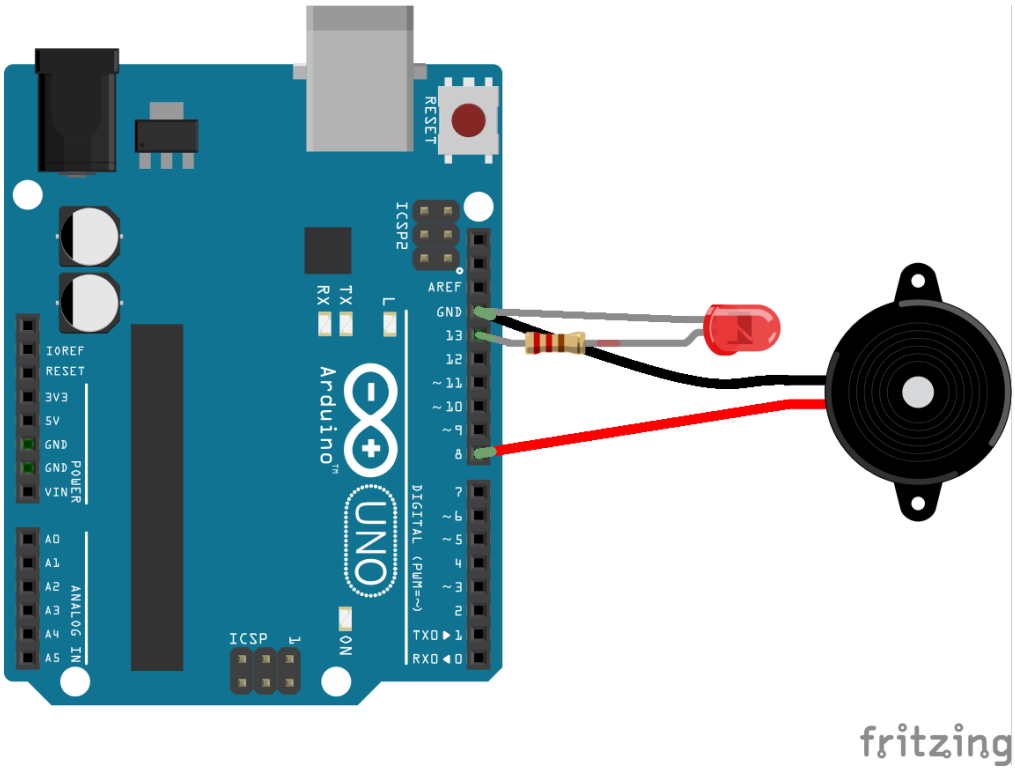


Figura 13 – Conexão física do nó com alertas físicos.

4.5 Nó IoT alarme de incêndio

Pensando na economia de energia de modo geral para o protótipo, ao invés de conectar os sensores diretamente no 5V do Arduino, os sensores foram conectados a portas digitais. Dessa forma, os sensores gastam energia apenas quando as portas digitais estão ligadas, sendo fácil programar a ligação e o desligamento das portas digitais.

Ao utilizar portas digitais, existe a necessidade de realizar uma pequena alteração relativo ao sensor de gás MQ-2, já que o mesmo gasta 160mA (Hanwei, 2018), porém uma placa Arduino UNO aceita apenas 20mA por porta digital. Assim, não há como energizar o sensor diretamente na porta digital, faz-se com que seja necessário o uso de um transistor NPN BC327 para o uso correto do sensor MQ-2.

Esse transistor faz a mesma função de uma porta digital, ou seja, de controle da passagem de energia. Com o uso do transistor, é possível fazer o controle da saída de voltagem (de 5V) do Arduino, através de uma porta digital.

Assim, todas as conexões físicas do nó sensor – incluindo os três sensores, antena XBee, Arduino UNO e avisos físicos, é exemplificada na Figura 14.

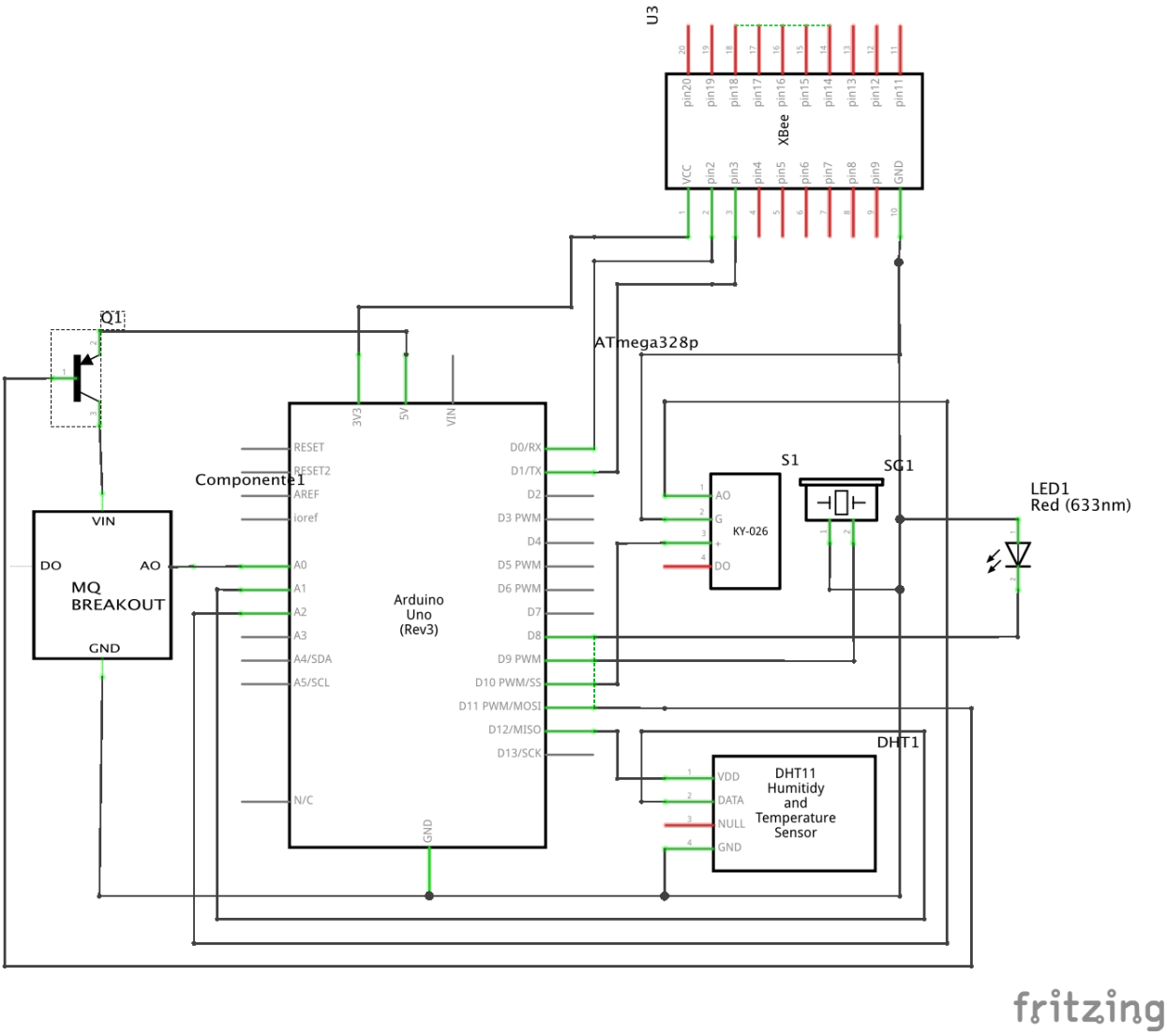


Figura 14 – Conexão física do nó IoT usando portas digitais.

5. GERENCIAMENTO DE DISPOSITIVO IOT COM *FOG*

A arquitetura utilizada no protótipo, envolvendo o dispositivo IoT, o nó *Mist* e a *Fog* é mostrada, em seu total, na Figura 15. Embora a figura esteja considerando uma arquitetura completa com quatro níveis (*Cloud*, *Fog*, *Mist* e nó IoT), o trabalho desenvolveu e focou apenas na troca de mensagens entre o nó *Mist* e o nó IoT.

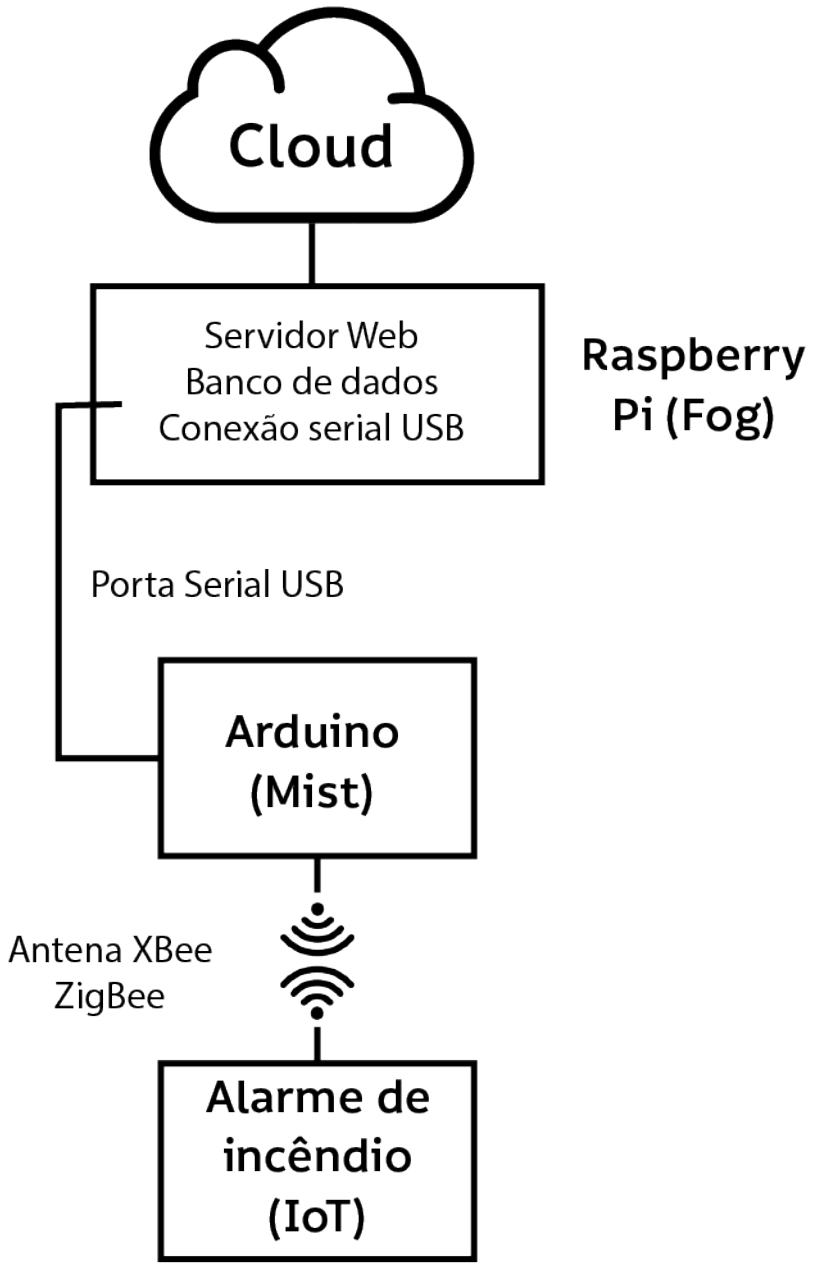


Figura 15 – Arquitetura utilizada no desenvolvimento deste trabalho.

5.1 Exibição dos possíveis estados do protótipo para um usuário

A condição de perigo é indicada com a combinação/cruzamento dos valores dos sensores do ar e gás, pois caso haja valor alto de fumaça - porém sem aumento de temperatura - pode indicar um fósforo que foi aceso, sem o acontecimento de um princípio de incêndio. Os possíveis estados do protótipo se encontram no Quadro 7.

Estados possíveis (Temperatura e umidade, Gás e fumaça, chamas)	Temperatura e Umidade	Gás e fumaça	Chamas	Condição Definida
(0, 0, 0)	Baixa	Baixo	Não	Condição normal
(1, 0, 0)	Alta	Baixo	Não	Abertura da porta do fogão, saindo ar quente
(0, 1, 0)	Baixa	Alto	Não	Vazamento de gás
(1, 0, 1)	Alta	Baixo	Sim	Erro de sensor (falso positivo)
(0, 0, 1)	Baixa	Baixo	Sim	Acendimento de fósforo
(0, 1, 1)	Baixa	Alto	Sim	Erro de sensor (falso positivo)
(1, 1, 0)	Alta	Alto	Não	Alimento queimando dentro do fogão
(1, 1, 1)	Alta	Alto	Sim	Incêndio

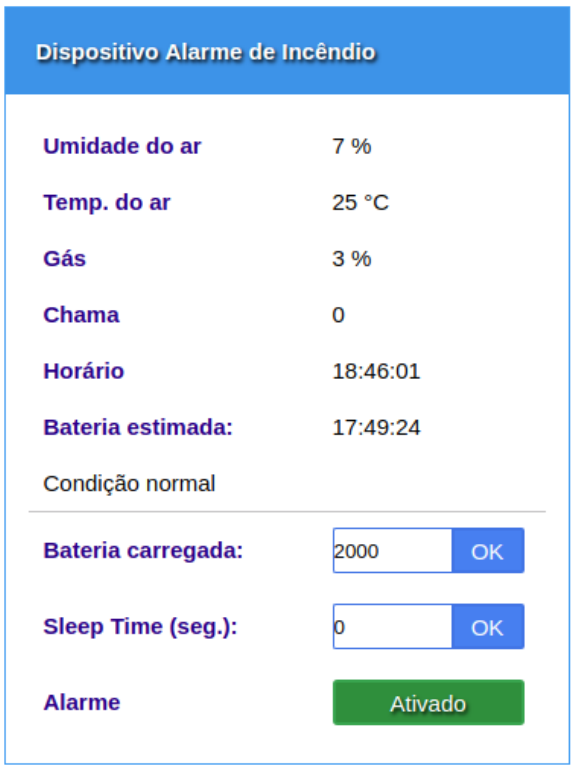
Quadro 7 – Possíveis estados do protótipo IoT.

Os valores definidos como limite para cada um dos sensores foram de 30ºC de temperatura, 4% de gás e fumaça e sem presença de chamas. Quaisquer valores acima desses são considerados valores “altos”, e apresentam diferentes cenários para cada um dos possíveis estados do protótipo.

Esses possíveis estados, e consequentemente as situações que eles representam, poderão ser acompanhados através do uso de um sistema desenvolvido pelo LRG (Laboratório de Redes e Gerência) da UFSC (Universidade

Federal de Santa Catarina). Dessa forma também é possível realizar ações como ligar ou desligar o alarme.

A Figura 16 mostra a exibição no site do LRG da situação (0, 0, 0) do alarme – conforme o Quadro 7 - ou seja, a condição detectada é considerada normal. Já a Figura 17 mostra a a exibição no site do LRG da situação (1, 0, 0), onde há um aumento na temperatura e umidade, porém com um valor de gás baixo e sem detecção de chama, ou seja, muito possivelmente a detecção da abertura da porta do fogão, com a saída do ar quente.



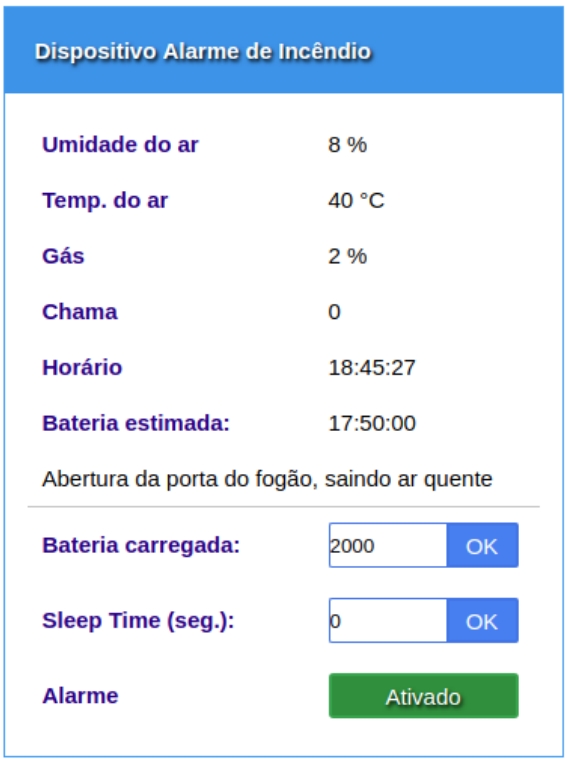


Figura 16 – Site do LRG exibindo a situação normal de funcionamento do alarme (0, 0, 0).

Figura 17 - Site do LRG exibindo a situação de abertura da porta de um fogão (1, 0, 0).

A Figura 18 mostra a exibição no site do LRG da situação (0, 1, 0), em que é detectado um vazamento de gás, devido ao alto valor capturado de gás, porém sem

a presença de chama ou aumento de temperatura e umidade. Já a Figura 19 mostra a exibição no site do LRG da situação (1, 0, 1), ou seja, onde existe a captura de altos níveis de temperatura e umidade, além da detecção de chama, porém os valores capturados de gás e fumaça são considerados baixos. Pode-se então, caracterizar essa situação como um erro do alarme, informando um falso positivo.

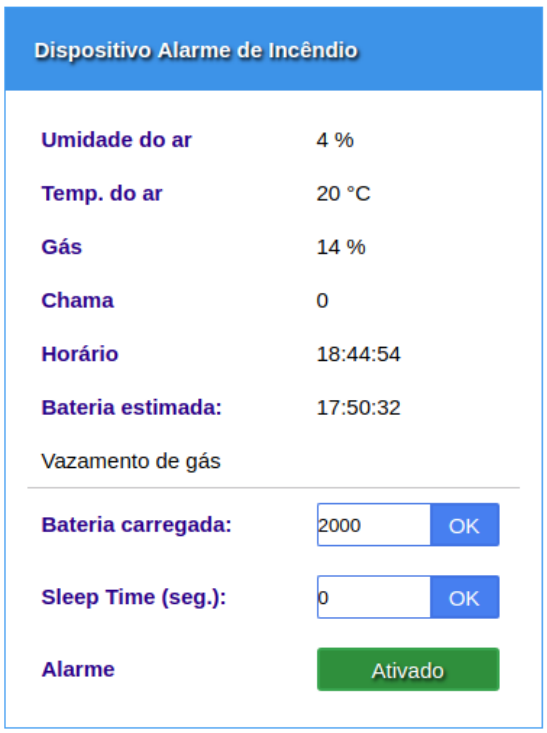


Figura 18 – Site do LRG exibindo a situação de vazamento de gás (0, 1, 0).

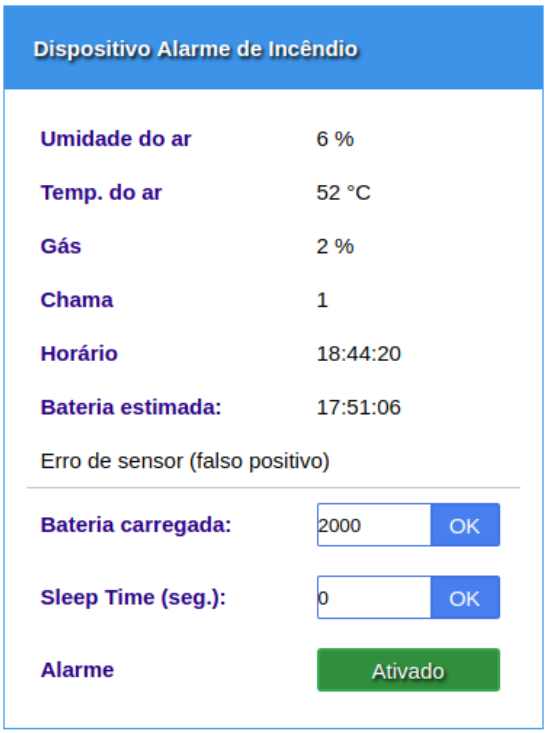


Figura 19 – Site do LRG exibindo a situação de falso positivo (1, 0, 1).

A Figura 20 mostra a exibição no site do LRG da situação (0, 0, 1), em que muito possivelmente há apenas o acendimento de um fósforo no ambiente no qual o alarme inteligente está localizado. Já a Figura 21 mostra a exibição no site do LRG da situação (0, 1, 1) – ou seja, em que novamente há a detecção de um falso positivo pelo alarme, dessa vez pelo valor baixo de umidade e temperatura, porém altos valores de gás e fumaça, assim como a presença de chamas.

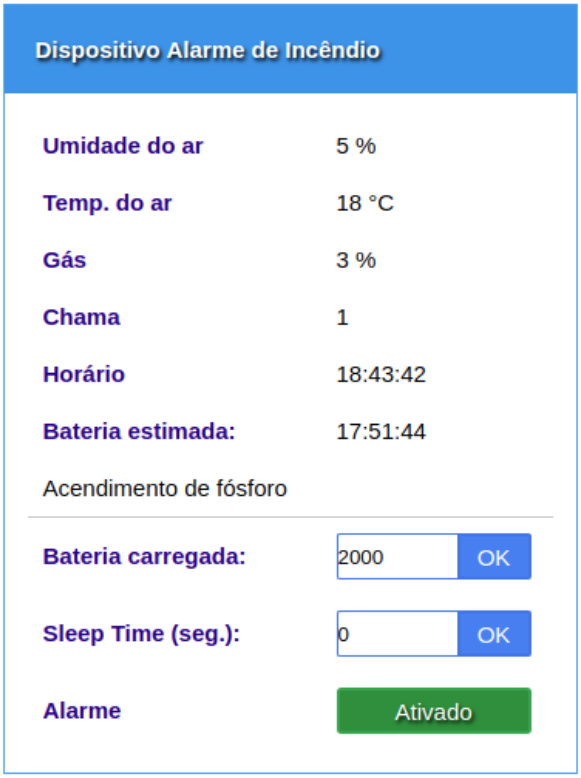


Figura 20 – Site do LRG exibindo a situação de acendimento de fósforo (0, 0, 1).

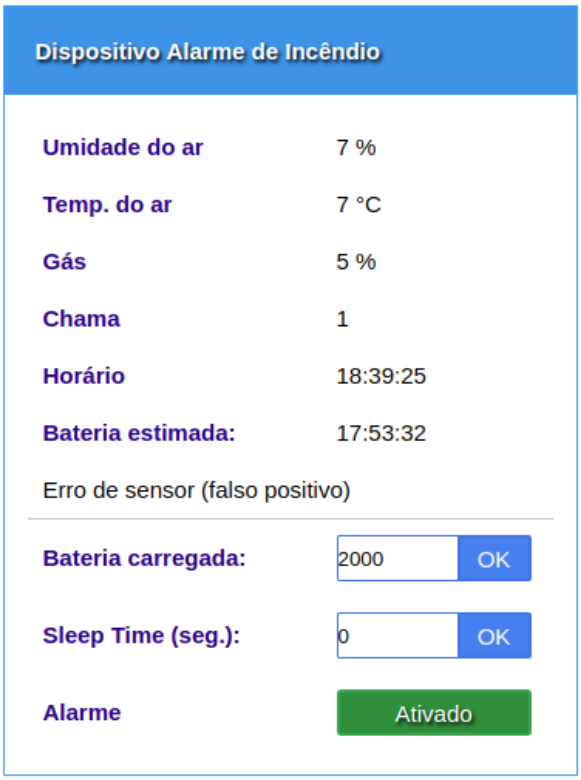


Figura 21 – Site do LRG exibindo a situação de falso positivo (0, 1, 1).

A Figura 22 mostra a exibição no site do LRG da situação (1, 1, 0), em que há a grande possibilidade de um alimento estar queimando dentro do fogão (considerando que haja um alarme de incêndio instalado na cozinha, devido ao grande risco de acidentes no local, quando comparando-se com o resto de uma casa), já que se tem uma situação com alta temperatura e umidade, altos valores de fumaça e gás, mas não é detectada a presença de chamas. Finalmente, a Figura 23 mostra a exibição no site do LRG da situação (1, 1, 1), ou seja, uma situação de emergência na qual há a detecção de um incêndio.

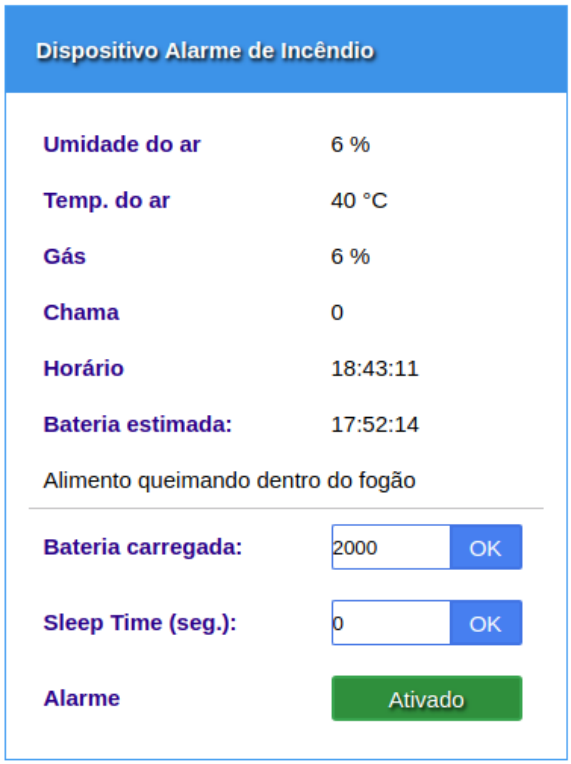


Figura 22 – Site do LRG exibindo a situação de alimento queimando dentro de um fogão (1, 1, 0).

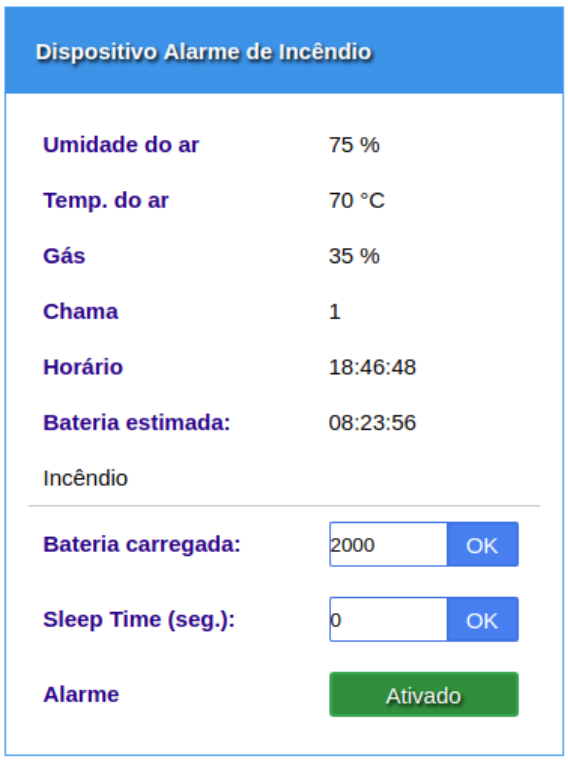


Figura 23 – Site do LRG exibindo a situação de incêndio (1, 1, 1).

5.2 Comunicação entre *Mist* e *Fog*

O conceito de *Fog Computing* é por natureza considerado mais leve e mais próximo do usuário do que a *Cloud*. Ainda assim, é possível usar um tipo de estrutura subsidiária da *Fog* – a *Mist Computing*. *Mist Computing* é definido como uma camada mais leve da *Fog*, com nós especializados, e principalmente com baixo poder computacional, mas ainda residindo dentro da rede de *Fog*. Esses nós, denominados "nós *Mist*" usam microcontroladores que alimentam nós da *Fog* (Iorga et al, 2018). No caso dos nós apresentados aqui, o nó coordenador pode ser considerado como nó *Mist*, devido ao baixo poder computacional do Arduino. Quanto ao nó sensor, esse é considerado apenas como um nó IoT.

Depois da captura de dados pelo nó IoT, esses dados são enviados para o nó *Mist* através da antena XBee. O nó *Mist*, por sua vez, encontra-se conectado a um Raspberry Pi, e realiza um pré-processamento dos dados (envio apenas do endereço do nó IoT e dados capturados, ao invés de um pacote inteiro padrão) antes de enviá-los ao Raspberry Pi (sendo ele o nó mais forte da *Fog*).

Para o envio e recebimento de pacotes entre a *Fog*, *Mist* e o nó IoT foi criado um protocolo novo especificamente para atender as necessidades dos dispositivos envolvidos no trabalho aqui desenvolvido. Esse protocolo foi baseado no protocolo ZigBee, embora ele esteja estruturado de forma que também seja possível realizar uma “tradução” de e para outros protocolos, como o Wi-Fi e o Bluetooth. A Figura 24 exemplifica este pacote reduzido, que é enviado ao Raspberry Pi pelo nó *Mist*.

1	2	3	4	5	6	7	8	9	10
11	AA	6B	32	35	37	34	39	39	31

Figura 24 – Pacote enviado ao Raspberry Pi pelo nó *Mist*.

A Figura 25 mostra com mais detalhes o pacote enviado ao Raspberry Pi. É necessário comentar também sobre a necessidade de envio de um Byte ID no pacote, para que o Raspberry Pi reconheça apenas pacotes que iniciam com o valor hexadecimal 11.

	Byte ID	Nó Origem		Umidade		Temperatura		Gás		Chamas
Byte	1	2	3	4	5	6	7	8	9	10
Valores (HEX)	11	AA	A8	37	30	32	30	30	34	30

Figura 25 – Detalhes do pacote enviado ao Raspberry Pi.

É importante notar que o envio para o Raspberry Pi é feito devido ao fato que o Raspberry Pi é o nó *Fog* centralizador da *Mist*.

Imaginando que na *Fog* de uma casa inteligente, seria possível que tivéssemos mais de um serviço na *Fog*, sendo que o serviço de alarme de incêndio para ambientes inteligentes seja um desse. Com isso, o uso do Raspberry Pi se torna importante, ao invés de realizar apenas o envio de dados diretamente para a *Cloud*.

Pode-se considerar que o protocolo de troca de mensagens do nó IoT (com os valores dos dados capturados pelos sensores) para o nó *Mist* e depois para a *Fog* é realizado, de forma resumida, segundo a Figura 26.

Figura 26 – Envio dos dados capturados pelo nó IoT para o nó Mist e para a Fog, de forma resumida.

A partir do envio para o Raspberry Pi, seria possível enviar dados para uma *Cloud*, mantendo um histórico dos dados capturados pelo nó sensor. Para os fins desta monografia, o envio de dados para a *Cloud* não é considerado, dado que o foco é apenas na *Mist*.

Tendo isto em mente, ainda é necessário que o usuário possa alterar alguns parâmetros do nó IoT, tal como desligar ou ligar os avisos físicos do mesmo (buzzer e LED), caso, por exemplo, haja um disparo acidental do alarme de incêndio. Nesse caso, o usuário irá realizar esse controle através do site em desenvolvimento no LRG,

escolhendo a função a ser realizada como de ativar ou desativar o alarme, ou alterar a frequência de envio dos dados capturados.

Os valores das funções, a nível de dados enviados no pacote, são descritos no Quadro 8.

Função	Decimal	HEX
Ligar/desligar avisos físicos	a	61
Alterar freqüência de envio de dados	b	62

Quadro 8 – Funções possíveis de envio da *Fog* para o nó *Mist*.

Dessa forma, será enviado um pacote do Raspberry Pi para o nó *Mist*, para que esse depois envie um pacote indicando o comando para o nó IoT e o mesmo realize a função desejada. A Figura 27 mostra um pacote enviado da *Fog* para o nó *Mist*, indicando que os avisos físicos do nó IoT devem ser desligados, enviando o valor “D” para indicar o desligamento.

	Byte ID	Função	Nó Destino		Valor
Bytes	1	2	3	4	5
Valores (HEX)	11	61	AA	A8	44

Figura 27 – Pacote enviado da *Fog* para o nó *Mist* requisitando o desligamento dos avisos físicos do alarme.

O mesmo acontece caso o usuário deseje ligar os avisos físicos, enviando o valor “L” para indicar a função de ligar. A Figura 28 mostra essa situação.

	Byte ID	Função	Nó Destino		Valor
Bytes	1	2	3	4	5
Valores (HEX)	11	61	AA	A8	4C

Figura 28 – Pacote enviado da *Fog* para o nó *Mist* requisitando o ligamento dos avisos físicos do alarme.

Ainda é possível que o usuário deseje alterar a frequência de envio de dados do nó sensor para um intervalor maior ou menor de tempo. Para exemplificar essa situação, a Figura 29 mostra um pacote enviado da *Fog* para o nó *Mist*, indicando a alteração da frequência de dados para 3 segundos.

	Byte ID	Função	Nó Destino		Valor
Bytes	1	2	3	4	5
Valores (HEX)	11	62	AA	A8	33

Figura 29 – Pacote enviado da *Fog* para o nó *Mist* requisitando que a frequência de envio de dados seja de 3 segundos.

5.3 Comunicação da *Mist* para o Nó IoT

Depois desse envio da *Fog* para o nó *Mist*, o nó *Mist* envia um pacote no modo API para o nó IoT, com o conteúdo de dados. No caso do desligamento dos avisos físicos, será enviado o valor “D” (para desligar). O mesmo pode ser feito para ligar os avisos físicos, porém com envio do dado “L” (para ligar). Um exemplo disso se encontra nas Figura 30 e 31, respectivamente.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
7E	00	0F	10	01	00	13	A2	00	40	D9	AA	A8	FF	FE	00	00	44	8D

Figura 30 – Pacote enviado para o nó sensor para desligar avisos físicos.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
7E	00	0F	10	01	00	13	A2	00	40	D9	AA	A8	FF	FE	00	00	4C	85

Figura 31 – Pacote enviado para o nó sensor para ligar avisos físicos.

Outra possibilidade ainda é a mudança de frequência de captura de dados pelo nó sensor. Isso pode ser realizado enviando um pacote do nó coordenador com o valor “F” para indicar mudança de frequência, e o valor em segundos do novo tempo

de captura. Por exemplo, o valor “F3” indicaria que a nova frequência de captura de dados pelo nó sensor seria a cada 3 segundos. Esse pacote se encontra exemplificado na Figura 32.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
7E	00	11	10	01	00	13	A2	00	40	D9	AA	A8	FF	FE	00	00	46	33	58

Figura 32 – Pacote enviado para o nó sensor para mudar a frequência de captura dos dados.

Dessa forma, o monitoramento dos dados capturados pelo nó IoT pode ser realizado de forma simples e rápida pelo usuário, graças a arquitetura de *Fog Computing* e sua integração com a *Cloud* (quando utilizando o site do sistema *Fog*).

O envio de um pacote da Fog para o nó Mist e que depois será enviado para o nó IoT pode ser exemplificado, de forma reduzida, na Figura 33.

	Byte ID	Função	Nó Destino		Valor
Bytes	1	2	3	4	5
Valores (HEX)	11	61	AA	A8	44

Pacote de dados da Fog para o nó mist, solicitando o desligamento do alarme

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
7E	00	0F	10	01	00	13	A2	00	40	D9	AA	A8	FF	FE	00	00	44	8D

Figura 33 – Pacote de dados enviado da Fog para o nó Mist, que será encaminhado para o nó IoT.

6. ESTIMATIVA AUTONÔMICA DE VIDA ÚTIL DO NÓ IOT

Neste capítulo serão apresentados os cálculos de estimativa de vida do nó IoT. Esses resultados dos valores de estimativa serão então exibidos para o usuário no sistema *Fog* (site do LRG), conforme os cálculos que serão apresentados nas seções 6.1 e 6.2.

6.1 Estimativa de gasto de energia em um ciclo

Para realizar a captura de dados do nós sensor sobre os valores de fumaça e gás, temperatura e umidade, assim como a existência de chamas, foi criado um programa usando a linguagem C para ser realizado o *upload* do mesmo no microcontrolador da placa Arduino UNO. O programa encontra-se no apêndice A.

Cada um dos sensores possui diferentes tempos de resposta - 2 segundos para o DHT11 (Aosong, 2018), 1 segundo para o MQ-2 e 1 segundo para o sensor de chama. Assim, cada sensor precisa ficar ligado ao menos durante o tempo de resposta, mas não há necessidade de gasto de energia quando os mesmos não estão enviando dados à *Fog*.

Para que a captura de um sensor não dependa do outro, e assim seja gasto menos energia, foi utilizado o conceito de *threads* para realizar a captura de dados através de um paralelismo pipeline. Com isso foi definido um ciclo mínimo de 2 segundos, devido a limitação do tempo de resposta do sensor DHT11.

Considerando-se que o sensor DHT11 gasta 0,3mA (Aosong, 2018), que o sensor MQ-2 gasta 160mA (Hanwei, 2018), e que o sensor de chama gasta 0,4mA, pode-se calcular que são gastos 160.7mA por hora. Com isso, podemos calcular o gasto de energia dos sensores, além do gasto do microprocessador do Arduino.

Para realizar o cálculo de gasto de energia no protótipo, é preciso utilizar os valores de cada um dos módulos utilizados no mesmo. Os valores de energia dos módulos, por hora e por segundo, se encontram no Quadro 9.

Módulo	Gasto por hora	Gasto por segundo
Umidade e Temperatura (DHT11)	0,3mA	0,00008333mA
Gás e Fumaça (MQ-2)	160mA	0,04444mA
Chamas	0,4mA	0,0001111mA
ATMEGA 328P do Arduino (sleep-mode)	0,1µA	0,000000027778mA
ATMEGA 328P do Arduino	3,2mA	0,00088889mA
Antena XBee (transmissão de dados)	33mA	0,0091666mA
Antena XBee (recebimento de dados)	28mA	0,00777778mA
Antena XBee (sleep-mode)	1µA	0,0000002778mA

Quadro 9 – Gasto de energia por módulo

Sabendo esses valores é possível calcular o ciclo do nó sensor como sendo:

$$Ciclo\ do\ nó\ sensor = captura\ dos\ sensores + gasto\ da\ antena + sleep - mode$$

Sendo que a captura de dados depende do tempo de resposta de cada um dos sensores mais a corrente, além do gasto de energia do Arduino propriamente dito. Para o cálculo do Arduino, o gasto enérgico é de 3,2mA por hora (Arduino, 2018). Considera-se que o Arduino ficará ligado apenas 2 segundos em um ciclo. Ou seja, o valor gasto na captura dos dados do nó sensor é:

$$Captura = (2s * 0{,}00008333mA) + (1s * 0{,}04444mA) + (1s + 0{,}000111mA) + (2 * 0{,}00088889)$$

$$Captura = 0{,}0465mA/ciclo$$

Após a captura dos dados pelo nó sensor, esses são enviados em um pacote até o nó coordenador. Para este calculo, foi considerado um pacote de tamanho médio de 25 bytes (ou 200 bits) e a velocidade de transmissão de 250kbps, a antena

gastaria 200/250000 segundos (ou seja, 0,0008 segundos) para o envio desse pacote. Com esse resultado devemos multiplicar o valor de energia gasto pela antena XBee para a transmissão de dados. Com esse calculo, temos o valor gasto de bytes por tempo.

$$Gasto\ da\ antena\ no\ modo\ de\ envio = 0{,}0008s * 0{,}0091666mA$$

$$Gasto\ da\ antena\ no\ modo\ de\ envio = 0{,}00000733328mA/pacote$$

Durante o tempo do ciclo de captura dos sensores, a antena fica ligada em modo recebimento, estando disponível para receber mensagens de controle da *Fog*. A antena, além do envio dos dados, também ficará ligada por um certo tempo durante o ciclo para recebimento dos dados. Assim, o gasto da antena XBee no modo de recebimento – explicitado no Quadro 9 – deve ser multiplicado pelo tempo do ciclo. Com isso, tem-se o valor de energia gasto pela antena XBee durante o recebimento dos dados.

$$Gasto\ da\ antena\ no\ modo\ de\ recebimento = 0{,}007777778mA/s * 2s$$

$$Gasto\ da\ antena\ no\ modo\ de\ recebimento = 0{,}015555556mA/ciclo$$

Considerando o tempo que o nó IoT ficará “acordado” em um ciclo, temos o valor de gasto energético em um ciclo como:

$$Gasto\ acordado = 0{,}062062889mA/ciclo$$

A definição do ciclo faz com que, depois do envio do pacote, não esteja sendo considerado uma mensagem de confirmação de recebimento antes do nó entrar em *sleep-mode.* Depois do envio do pacote, o nó IoT entra em *sleep-mode.* Isso é calculado com os valores do microprocessador e da antena XBee, ambos em *sleep-mode*.

$$Sleep - mode = (0{,}00000002778mA\ +\ 0{,}0000002778\text{mA}) * \text{T}$$

O tempo T do *Sleep-mode* é uma variável gerenciável a ser controlada na *Fog*, de modo a alterar o intervalo do ciclo, podendo ser modificado com o uso de mensagens enviadas para o nó IoT. É preciso ainda comentar que no início do ciclo, a antena XBee é ligada, e a mesma fica no modo recebimento durante o tempo do ciclo. No fim do ciclo é feito o envio dos dados para a *Fog*, e então tanto a antena quanto o Arduino entram em *sleep-mode*. A Figura 34 demonstra essa situação.

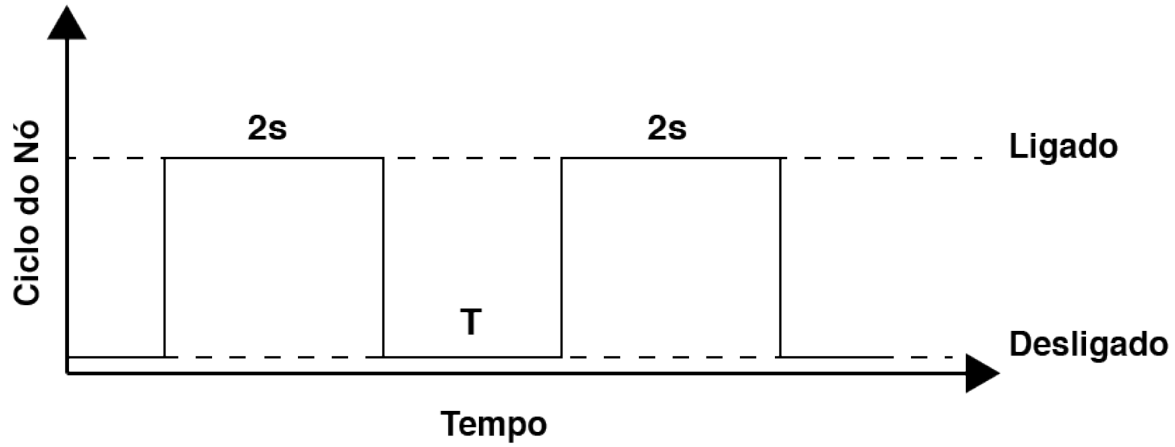


Figura 34 – Gasto de energia do nó IoT no tempo

Existe, porém, um limite associado ao tempo máximo de T. Segundo a instrução normativa IN 012/DAT/CBMSC do Corpo Militar de Bombeiros Estadual de Santa Catarina, foi definido que dispositivos utilizados para alarmes de incêndio possuem um tempo máximo de resposta para ser aceito dentro dos padrões de segurança estabelecidos em lei. O tempo de resposta máximo para a detecção de chamas por um dispositivo de alarme de incêndio é de 5 segundos (Santa Catarina, 2014). Tendo em mente esse valor, e considerando que o tempo já existente de duração de um ciclo do protótipo IoT é de 2 segundos, o tempo máximo em que o nó pode se encontrar em *sleep-mode* é de 3 segundos. O Quadro 10 faz uma comparação de estimativas de gasto de energia, alterando os valores de tempo T de *sleep-mode* do microprocessador do Arduino e da antena XBee.

	0 segundos	1 segundo	2 segundos	3 segundos
Gasto em sleep-mode	0mA	0,00000030556mA	0,00000061116mA	0,00000091667mA
Gasto acordado	0,062062889mA	0,062062889mA	0,062062889mA	0,062062889mA
Estimativa de gasto total	0,062062889mA/ciclo	0,06206319444mA/ciclo	0,06206350000mA/ciclo	0,06206380556mA/ciclo

Quadro 10 – Gasto energético alterando o valor T de *sleep-mode* do nó IoT

Outro valor importante de ser calculado é em relação ao tempo de vida de uma bateria para utilizar o protótipo. Como o foco é a economia de energia, o melhor então seria usar uma bateria como forma de alimentação, ao invés de utilizar o alarme de incêndio ligado sempre na corrente elétrica de uma residência. Baterias de fácil acesso no mercado possuem cerca de 2.000mA, podendo também variar entre 2.100 e 2.500mA cada. O cálculo para estimativa de ciclos que cada bateria consegue aguentar é a seguinte:

$$Estimativa\ de\ ciclos\ por\ bateria = potência\ bateria/(gasto\ ciclo)$$

Já a estimativa de tempo em horas, por bateria é calculada da seguinte forma:

$$Estimativa\ de\ vida\ em\ horas = \frac{Estimativa\ de\ ciclos\ por\ bateria * tempo\ do\ ciclo}{3600}$$

O Quadro 11 apresenta a quantidade de ciclos que cada bateria aguenta com o gasto energético do nó IoT, variando também o tempo de *sleep-mode*.

	Tempo de vida em ciclos			Tempo de vida em horas		
Tempo T de sleep-mode	**Bateria 2.000mA**	**Bateria 2.100mA**	**Bateria 2.500mA**	**Bateria 2.000mA**	**Bateria 2.100mA**	**Bateria 2.500mA**
0 segundos	32.225,377	33.836,645	40.281,721	17,902	18,798	22,378
1 segundo	32.225,218	33.836,479	40.281,523	26,854	28,197	33,567
2 segundos	32.225,059	33.836,312	40.281,324	35,805	37,595	44,757
3 segundos	32.224,901	33.836,146	40.281,126	44,756	46,994	55,946

Quadro 11 – Tempo de vida do nó IoT, usando diferentes baterias e considerando os gastos energéticos do protótipo IoT.

Os valores resultantes do tempo de vida do nó estarão disponíveis para exibição para o usuário através do site de sistema de monitoramento em desenvolvimento no LRG. Por exemplo, a Figura 35 mostra o resultado dos cálculos caso fosse utilizado 1 segundo como tempo de *sleep-mode*, utilizando uma bateria com potência de 2.000 mA. Já a Figura 36 mostra o resultado dos cálculos de tempo de vida do nó quando utilizando um tempo de *sleep-mode* de 2 segundos, também utilizando uma bateria com potência de 2.000 mA.

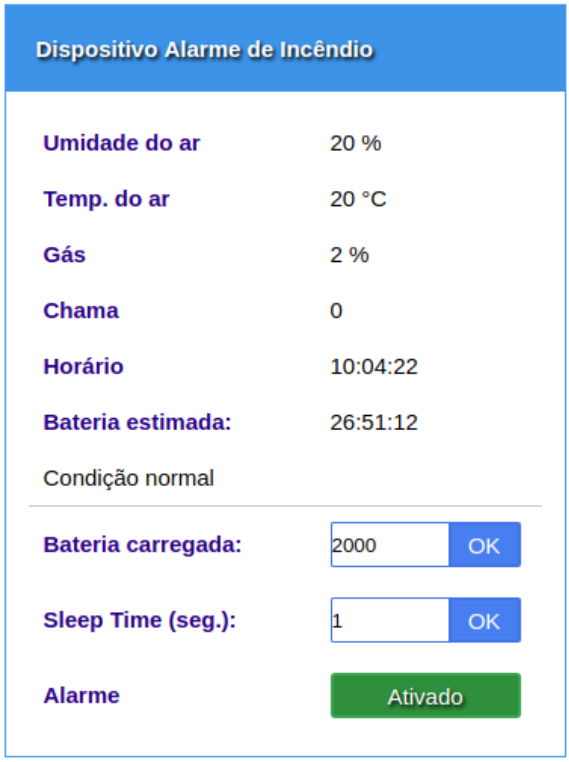


Figura 35 – Cálculos de tempo de vida exibidos no site do LRG, usando 1 segundo de *sleep-mode*

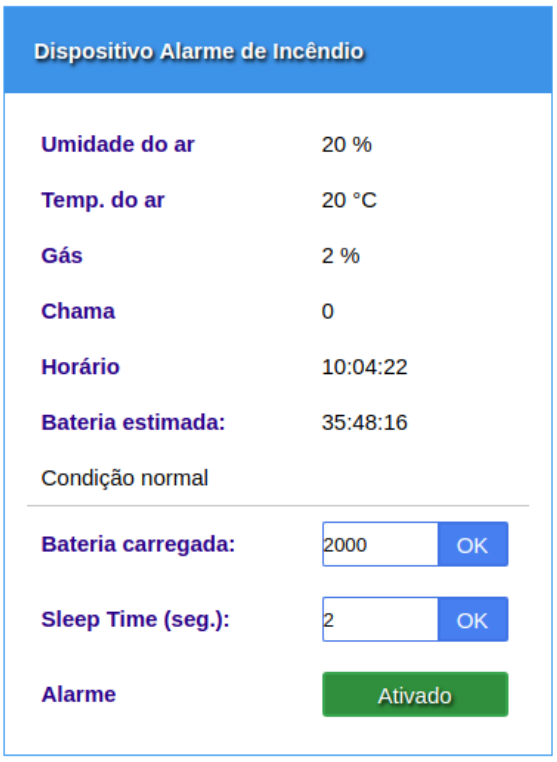


Figura 36 - Cálculos de tempo de vida exibidos no site do LRG, usando 2 segundos de *sleep-mode*

Finalmente, a Figura 37 mostra o resultado dos cálculos de tempo de vida do nó quando utilizando um tempo de *sleep-mode* de 3 segundos, também utilizando uma bateria com potência de 2.000 mA.

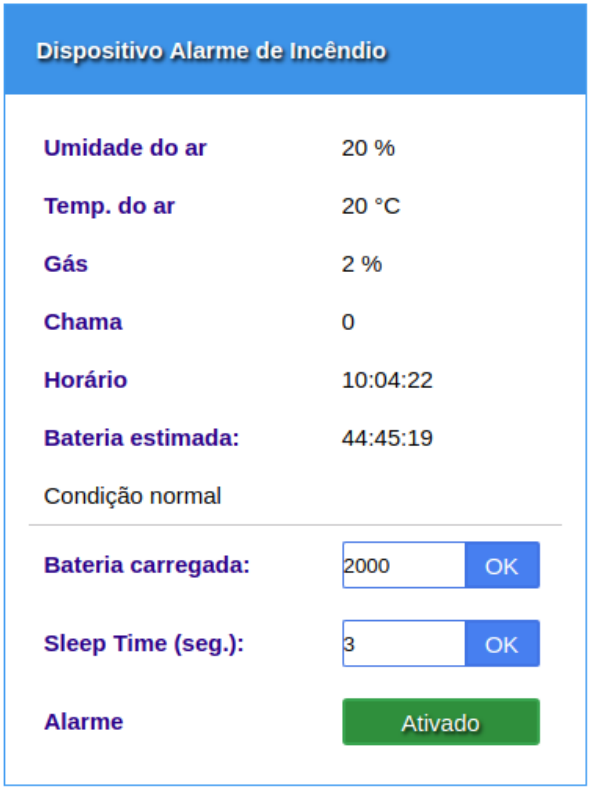


Figura 37 - Cálculos de tempo de vida exibidos no site do LRG, usando 3 segundos de *sleep-mode*

É possível notar uma grande diferença de tempo de vida das baterias no período quando se utiliza o tempo de *sleep-mode* do nó IoT, porém ainda assim os resultados ainda não são considerados ótimos – nos três tipos de baterias utilizadas, nenhuma delas consegue ter tempo de vida acima de três dias. Pode-se afirmar que isso provêm do gasto excessivo do sensor MQ-2, um valor acima mesmo do gasto energético da antena XBee.

Seria interessante então modificar o protótipo IoT para utilizar combinações diferentes de sensores. A título de brevidade, essas opções se encontram exemplificadas no Quadro 12 utilizando um tempo T de *sleep-mode* igual a 3 segundos.

Sensores	Tempo de vida em ciclos			Tempo de vida em horas		
	Bateria 2.000mA	Bateria 2.100mA	Bateria 2.500mA	Bateria 2.000mA	Bateria 2.100mA	Bateria 2.500mA
DHT11 e MQ-2	32.282,695	33.896,830	40.353,369	44,837	47,078	56,046
MQ-2 e sensor de chamas	32.311,671	33.927,254	40.389,589	44,877	47,121	56,096
DHT11 e sensor de chamas	113.511,493	119.187,068	141.889,367	157,654	165,537	197,068
Sensor de chamas	114.595,485	120.325,260	143.244,357	159,160	167,118	198,950

Quadro 12 – Opções de módulos no nó IoT para diminuição do gasto energético.

Pode-se perceber o quanto a presença do sensor MQ-2 influencia no gasto de energia quando presente no nó IoT. Uma possibilidade que permitira continuar a utilizar um sensor de gás (ao invés de utilizar apenas os sensores de chamas e DHT11) seria fazer uso de outro sensor de gás (como o MQ-9, que captura GLP, gás metano e monóxido de carbono), que tivesse um valor de gasto de energia menor do que o MQ-2.

6.2 Modo de emergência do alarme

As estimativas anteriores se baseiam em situações em que não era detectado fogo, ou seja, não eram situações de emergência. Para considerar uma situação de emergência, é preciso que o comportamento dos sensores seja diferente. Primeiramente, em uma situação de emergência (ao contrário de uma situação considerada normal de funcionamento), é preciso que os avisos físicos do alarme de incêndio (buzzer e LED) sejam ligados para servirem como forma de comunicação mais concreta a um usuário. Além disso, é preciso que todos os sensores do nó IoT fiquem ligados durante o tempo da situação de emergência – não apenas durante os 2 segundos do ciclo, como definido anteriormente (logo não há ciclo, apenas o consumo contínuo por segundo) – para que a captura dos dados seja feita

constantemente durante a situação de emergência, até que o usuário envie uma mensagem para desligar os alarmes físicos – ou seja, quando o nó IoT recebe uma mensagem da *Fog* para desligar os alarmes, como exemplificado na Fig 30. No pior dos casos, a bateria do alarme será consumida totalmente (no caso do usuário não desligar o alarme de incêndio). Esse intervalo da situação de emergência será chamado de TD.

Com esse envio constante dos dados, tem-se um valor diferente de pacotes enviados para a *Fog*. Em uma situação normal, apenas um pacote é enviado por ciclo, ou seja, um pacote a cada 2 segundos. Em uma situação de emergência, para acompanhamento da situação, pode-se enviar 1 pacote por segundo se o tempo TD for definido como 1.000 milissegundos (ou seja, 1 segundo). Para tempos menores de TD, seria melhor realizar o envio de mais pacotes. Com o tempo de 1.000 milissegundos como base, podemos calcular o número de pacotes enviado – e consequentemente o gasto de energia do nó – em uma situação de emergência, a cada 100, 200, 500 milissegundos, e assim por diante. Cada um dos tempos apresentará uma quantidade de pacotes diferente. Dessa forma, é possível diminuir o tempo de resposta e realizar um monitoramento da situação em tempo real. Considerando as limitações de tempo real, o envio de múltiplos pacotes entre a *Fog* e nó IoT é feito de forma a minimizar o tempo de resposta entre os dois.

O envio constante de pacotes é realizado na busca da garantia de entrega de pacotes. Ou seja, no evento de perda de um pacote, a inundação de pacotes na situação de emergência consegue indicar ao usuário que o nó IoT ainda se encontra ativo, e que ainda se tem uma situação de emergência sendo detectada pelo mesmo.

Os valores de gasto energético dos avisos físicos do dispositivo IoT (buzzer e LED) se encontram no Quadro 13.

Módulo	Gasto por hora	Gasto por segundo
Buzzer	25mA	0,00694444mA
LED	20mA	0,00555556mA

Quadro 13 – Gasto de energia de avisos físicos do nó IoT.

Com isso, é possível calcular uma estimativa de vida de bateria do dispositivo IoT quando em uma situação de emergência, variando o tempo TD, e o quanto de gasto energético tem-se a mais nesse caso, quando comparado com o funcionamento normal do dispositivo IoT. Nota-se que os outros valores utilizados no cálculo, como valores de gasto de energia dos sensores e da antena XBee ZigBee continuam com os mesmos valores do Quadro 8. O cálculo de gasto energético do dispositivo em situação de emergência é realizado inicialmente com o tempo necessário de envio dos pacotes pela antena:

$$Tempo\ de\ envio\ dos\ pacotes = 0{,}0008s * número\ de\ pacotes$$

Com esse valor, é possível calcular o gasto da antena no modo de transmissão para o envio do número de pacotes selecionados:

$$Gasto\ em\ modo\ de\ transmissão = gasto\ da\ antena\ em\ modo\ de\ tranmissão * tempo\ de\ envio\ dos\ pacotes$$

Depois disso, é calculado o gasto da antena no modo de recebimento de dados (pois a antena espera receber um pacote da *Fog* para desligamento do alarme. Isso significa, por exemplo, no caso do tempo TD ser 1 segundo que a cada segundo, tem-se 16 milissegundos para enviar dados para a *Fog*, e 984 milissegundos para ficar em modo de recebimento). O cálculo é feito da seguinte forma:

$$Gasto\ no\ modo\ de\ recebimento = gasto\ da\ antena\ em\ modo\ de\ recebimento * (1 - tempo\ de\ envio\ dos\ pacotes)$$

Com esse resultado, deve-se calcular o gasto da antena total, quando em uma situação de emergência:

$$Gasto\ da\ antena\ em\ modo\ de\ emergência = gasto\ no\ modo\ de\ recebimento + gasto\ no\ modo\ de\ transmissão$$

Finalmente, é possível calcular o gasto do nó IoT como um todo quando em situação de emergência. Isso é feito através do seguinte cálculo:

$$Gasto\ do\ nó\ IoT\ em\ situação\ de\ emergência = gasto\ sensores + gasto\ buzzer + gasto\ LED + gasto\ da\ antena\ em\ modo\ de\ emergência$$

Os resultados dos cálculos se encontram no Quadro 14. Foi considerada uma bateria de 2.000mA para a estimativa do tempo de vida do dispositivo IoT, com essa estimativa sendo em horas, caso não houvesse desligamento do alarme (ou seja, caso a bateria fosse usada apenas para o funcionamento do nó em situação de emergência).

Tempo TD em milissegundos	Número de pacotes/s	Gasto do nó em emergência	Acréscimo de gasto	Tempo de vida em horas de bateria de 2.000mA
50	20	0.06582778mA/s	112,132%	8,438
100	10	0,06581667mA/s	112,096%	8,440
200	5	0,06581111mA/s	112,078%	8,441
500	2	0,06580778mA/s	112,068%	8,442
1.000	1	0,06580667mA/s	112,064%	8,442

Quadro 14 – Gasto de energia do nó em situação de emergência.

Em geral, pode-se afirmar que o gasto do nó IoT em uma situação de emergência gasta cerca de 112% a mais de energia do que durante o funcionamento normal do mesmo, e que uma bateria de 2.000mA poderia durar cerca de 8 horas caso fosse utilizada apenas durante a situação de emergência.

Assim como no funcionamento “normal” do nó IoT, a vida útil estimada restante do nó é exibida no site da LRG da UFSC. Na Figura 38 é possível ver a exibição do tempo de vida restante do nó IoT, quando usando uma bateria de 2.000mA. É importante notar também que é exibida a situação de incêndio detectada pelos sensores.

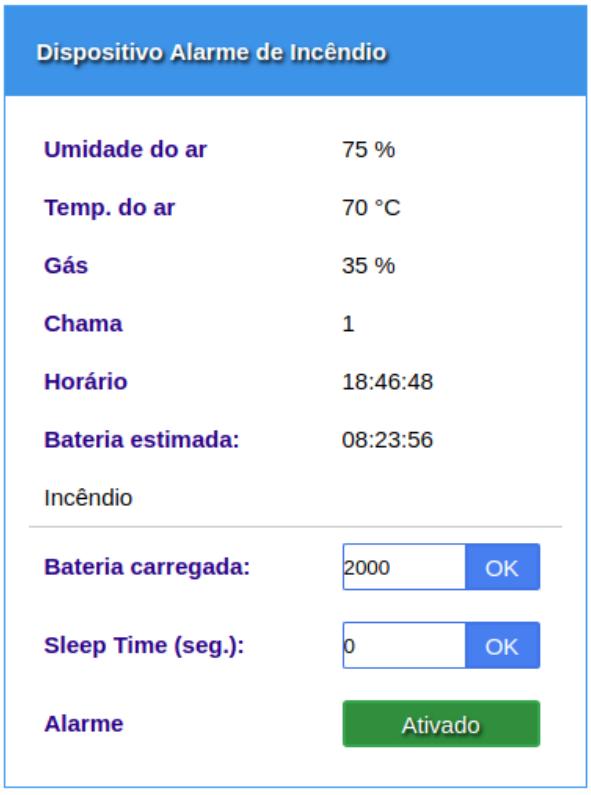


Figura 38 - Cálculos de tempo de vida exibidos no site do LRG quando em situação de emergência

6.3 Comparação de gasto de energia usando outros protocolos de rede

Nos cálculos de energia realizados anteriormente, foi considerado que a antena utilizada no nó IoT seria a antena XBee, que por sua vez utiliza o protocolo ZigBee para envio dos dados do nó IoT. Porém, também seria possível utilizar outros protocolos de rede de uso comum, tal como Wi-Fi e Bluetooth; para isso, seria necessário substituir a antena XBee pela antena XBee Wi-Fi ou pelo módulo Bluetooth 4.0 BLE, respectivamente.

O uso de diferentes protocolos de rede para envio de dados do nó IoT torna-se ainda mais difícil se for considerada a possibilidade de que existam mais serviços inteligentes em uma residência, que também utilizem outros protocolos de rede para executar suas funções. Dessa forma, é preciso se atentar ao gerenciamento da *Fog*, pois quanto mais protocolos e serviços diferentes em uma mesma residência, mais complexo ficaria o sistema em relação ao gerenciamento da *Fog*.

Por exemplo, em uma casa pode se ter mais de um tipo se serviço, como sistema de iluminação inteligente, juntamente com o sistema de alarme de incêndio inteligente proposto neste trabalho. Isso pode ser exemplificado através da Figura 39.

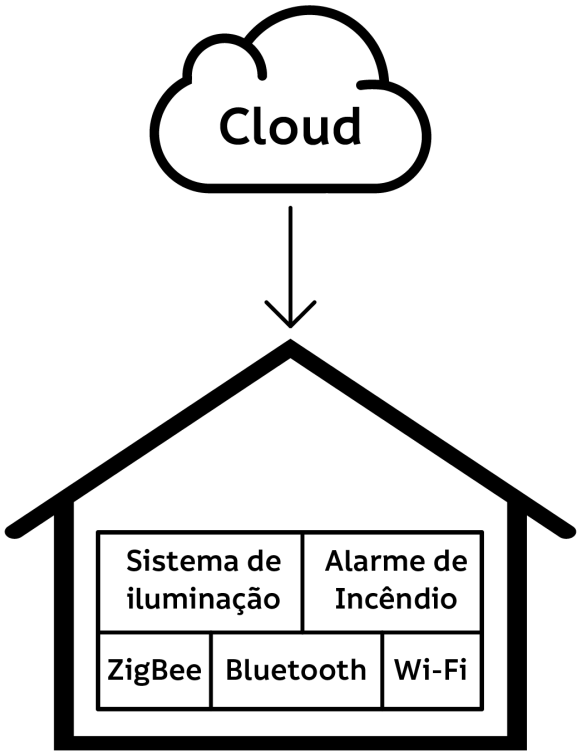


Figura 39 – Serviços inteligentes em uma residência, utilizando diferentes protocolos de rede.

A antena XBee Wi-Fi permite envio de dados via Wi-Fi, podendo enviar dados com velocidade de até 72 Mbps (Digi, 2018). Assim como a antena XBee ZigBee, também é possível realizar a configuração da mesma através do *software* XCTU. Os valores de gasto de energia quando utilizando a antena XBee Wi-Fi se encontra no Quadro 15.

Módulo	Gasto por hora	Gasto por segundo
Antena XBee Wi-Fi (transmissão de dados)	309mA	0,08583333mA
Antena XBee Wi-Fi (recebimento de dados)	100mA	0,02777777mA
Antena XBee Wi-Fi (sleep-mode)	6µA	0,00000166666mA

Quadro 15 – Gasto de energia da antena XBee Wi-Fi.

Os outros valores envolvidos no cálculo (como gasto do microcontrolador do Arduino e dos sensores) ainda continua o mesmo. Os gastos de energia do nó IoT, caso fosse utilizada a antena XBee Wi-Fi se encontram no Quadro 16, variando os

valores de tempo de *sleep-mode* do nó IoT e a potência da bateria utilizada. Para a estimativa de tempo de vida do nó IoT, foi levada em consideração apenas a situação de funcionamento normal do alarme (ou seja, não foi levada em consideração a situação de emergência).

	Tempo de vida em ciclos			Tempo de vida em horas		
Tempo T de sleep-mode	Bateria 2.000mA	Bateria 2.100mA	Bateria 2.500mA	Bateria 2.000mA	Bateria 2.100mA	Bateria 2.500mA
0 segundos	19.583,992	20.563,192	24.479,990	10,879	11,423	13,599
1 segundo	19.583,667	20.562,850	24.479,584	16,319	17,135	20,399
2 segundos	19.583,342	20.562,509	24.479,178	21,759	22,847	27,199
3 segundos	19.583,017	20.562,168	24.478,772	27,198	28,558	33,998

Quadro 16 – Tempo de vida de baterias de diferentes potências, utilizando antena XBee Wi-Fi.

Em comparação com o uso da antena XBee ZigBee, o uso do Wi-Fi para envio dos dados não tem valores melhores a oferecer – pelo contrário, o tempo de vida das diferentes baterias é menor do que se comparado com o uso do protocolo ZigBee.

Com o uso do módulo Bluetooth 4.0 BLE é possível realizar o envio dos dados capturados pelo nó IoT com o protocolo Bluetooth, porém sua grande desvantagem é a distancia permitida pelo módulo, de apenas 100 metros dentro de um ambiente fechado (Osoyoo, 2016). Ao contrário das antenas XBee ZigBee e Wi-Fi, o módulo Bluetooth 4.0 BLE não possui diferenciação de gasto de energia quanto a envio e recebimento de dados. O Quadro 17 mostra o gasto energético do módulo.

Módulo	Gasto por hora	Gasto por segundo
Bluetooth 4.0 BLE	8.5mA	0,00236111mA
Bluetooth 4.0 BLE (sleep-mode)	1,5mA	0,000416mA

Quadro 17 – Gasto de energia do modulo Bluetooth 4.0 BLE.

Assim, como na estimativa realizada com o uso da antena XBee Wi-Fi, os outros valores envolvidos no cálculo, como o gasto do microcontrolador do Arduino e

dos sensores, ainda continuam os mesmos. Os gastos de energia do nó IoT, caso fosse utilizado o módulo Bluetooth 4.0 BLE se encontra no Quadro 18, variando os valores de tempo de *sleep-mode* do nó IoT e a potência da bateria utilizada. Para a estimativa de tempo de vida do nó IoT, foi levada em consideração apenas a situação de funcionamento “normal” do alarme (ou seja, não foi levada em consideração a situação de emergência).

	Tempo de vida em ciclos			Tempo de vida em horas		
Tempo T de sleep-mode	**Bateria 2.000mA**	**Bateria 2.100mA**	**Bateria 2.500mA**	**Bateria 2.000mA**	**Bateria 2.100mA**	**Bateria 2.500mA**
0 segundos	39.045,553	40.997,830	48.806,941	21,691	22,776	27,114
1 segundo	38.730,479	40.667,003	48.413,099	32,275	33,889	40,344
2 segundos	38.420,449	40.341,472	48.025,562	42,689	44,823	53,361
3 segundos	38.115,344	40.021,111	47.644,180	52,937	55,584	66,172

Quadro 18 - Tempo de vida de baterias de diferentes potências, utilizando o módulo Bluetooth 4.0 BLE.

Quando comparado com os valores obtidos na estimativa de gasto de energia utilizando a antena XBee ZigBee, o uso do módulo Bluetooth 4.0 BLE, e consequentemente o uso do protocolo de rede Bluetooth, apresenta uma maior economia de energia das baterias – ou seja, um maior tempo de vida das baterias, em todos os casos. Isso se deve principalmente ao fato de o módulo não possuir diferentes valores energéticos nos casos de envio e transmissão de dados (como ocorre em ambas antenas XBee ZigBee e Wi-Fi), além de valores menores de gasto no módulo quando acordado.

7. CONCLUSÕES E TRABALHOS FUTUROS

Como os níveis de poluição relacionados a comunicação e tecnologia devem crescer exponencialmente nos próximos anos devido ao uso cada vez mais cotidiano de tecnologias como a *Internet of Things*, uma das preocupações que se deve ter em mente é o gasto de energia que as mesmas acabam tendo. Dessa forma é preciso saber quais soluções podem ser tomadas, principalmente tomando como principio o uso de dispositivos de *Internet of Things*, utilizando os mesmos em um contexto de casas inteligentes e fazendo uso de uma arquitetura de *Fog Computing.*

Com esses conceitos em mente, foi criado um protótipo IoT de alarme de incêndio, onde é possível realizar a captura de dados de fumaça, gás, umidade, temperatura e presença de chamas, e enviar os mesmos para um nó na *Mist*, e consequentemente, para a *Cloud* posteriormente. Assim, é possível realizar o monitoramento em tempo real de uma situação caso seja detectado um cenário de incêndio na residência do usuário.

Quanto a necessidade de economia de energia, foi decidido que o nó IoT usaria uma bateria (ao invés de ser ligado à corrente elétrica da residência) para seu funcionamento. Além disso, foram realizados cálculos do tempo de vida útil dessa bateria, alterando parâmetros de potência da bateria e diferentes protocolos de rede utilizados para realizar o envio dos dados dentro da arquitetura proposta.

Com esses resultados, pode-se notar que o tempo de vida de uma bateria, quando utilizando os sensores DHT11, MQ-2 e sensor de chama, e quando utilizando a antena XBee ZigBee, não conseguiria durar muito tempo (no melhor dos casos, a bateria precisaria ser trocada a cada dois dias). Isso se deve principalmente ao fato de que o sensor de fumaça e gás MQ-2 possui um gasto energético muito alto – maior mesmo do que o gasto da antena XBee ZigBee.

Dessa forma, uma opção para aumento do tempo de vida do nó IoT seria trocar o uso do sensor MQ-2 por outro sensor de gás e fumaça que tenha um valor de gasto energético menor do que o MQ-2.

Outra opção para economia de energia seria utilizar o nó IoT ligado na corrente elétrica para o funcionamento “comum”, e utilizar a bateria apenas para uma situação de falta de energia elétrica. Assim, ainda seria possível utilizar o alarme de incêndio mesmo numa situação de queda de energia.

Quanto a comparação de tempo de vida do nó quando utilizando outros protocolos de rede (Wi-Fi e Bluetooth), é possível perceber que o uso do Wi-Fi não é recomendado para ser o protocolo de envio de dados no caso do protótipo de nó IoT desenvolvido neste trabalho; Devido ao grande gasto de energia da antena XBee Wi-Fi, o tempo de vida do nó diminui ainda mais do que se fosse utilizada a antena XBee ZigBee. Dessa forma, o uso do protocolo de Wi-Fi não seria recomendado.

Quanto ao uso do Bluetooth, os resultados do tempo de vida foram melhores quando comparados com os valores de uso com ZigBee. Porém, vale ressaltar que a distância física possível dentro da residência é menor quando usando ZigBee do que quando usando o Bluetooth. O que deve ser considerado nesse caso também, é que os valores estimados são em relação a dois módulos específicos, e que com outras opções de peças esses valores podem mudar (principalmente em relação ao alcance de cada um dos módulos).

7.1 Principais Contribuições

As principais contribuições deste trabalho foram:

- Projeto de hardware do nó IoT identificando as conexões físicas dos sensores, atuadores, e antena;

- Protocolo de comunicação da *Mist* (comunicação da *Mist* para a *Fog* e comunicação com o nó IoT);
- Método de cálculo de estimativa de tempo de vida útil do nó IoT.

7.2 Trabalhos futuros

Uma questão que pode ser vista em trabalhos futuros de modo a otimizar a economia de energia do nó IoT é utilizar uma forma de contornar o problema de gasto excessivo de energia relativo ao uso do MQ-2 no nó IoT. Assim, seria utilizar outro sensor para realizar o mesmo trabalho, porém com um gasto de energia menor do que o MQ-2, como por exemplo o MQ-9, que gasta menos energia. Nesse caso, deve-se atentar a quais dados o sensor pode detectar – no caso do MQ-9, ele consegue capturar GLP, gás metano e monóxido de carbono, porém não possui capacidade de detectar fumaça, os gases butano e propano, nem mesmo álcool ou hidrogênio, como é o caso do MQ-2.

Além disso, também poderia ser alterado a forma de recebimento de dados do nó IoT, após o recebimento de pacotes vindos do nó *Mist*. No protótipo atual, a antena é ligada (em modo de recebimento) no início do ciclo, e então entra em *sleep-mode* no final do ciclo. Ao invés desta abordagem, seria possível fazer com que a antena ficasse em *sleep-mode* durante a grande maioria do ciclo, e fosse ligada apenas após a captura dos dados pelos sensores, para que a antena possa esperar um pacote da *Fog* durante um certo tempo – por exemplo, 100 milissegundos. Nesses 100 milissegundos, devem ser considerados o tempo de envio de um pacote da antena para a *Fog* (pelo nó *Mist*), o tempo de processamento da *Fog*, e o tempo de envio de um pacote da *Fog* para a antena. Dessa forma, seria possível realizar uma economia de energia que certamente ajudaria a aumentar o tempo de vida útil do nó IoT

Referências Bibliográficas

ABREU, Ricardo; RODRIGUES, Matheus; TELES, Lilia; TOLEDO, Nathalia; TORRES, Lívia. **Incêndio de grandes proporções destrói o Museu Nacional, na Quinta da Boa Vista**. *G1*, Rio de Janeiro, 2 de Setembro de 2018. Disponível em: <https://g1.globo.com/rj/rio-de-janeiro/noticia/2018/09/02/incendio-atinge-a-quinta-da-boa-vista-rio.ghtml>. Acesso em 11 de Outubro de 2018.

ARDUINO. **Arduino UNO Rev 3**. 2018. Disponível em: <https://store.arduino.cc/arduino-uno-rev3>. Acesso em 2 de Julho de 2018.

AOSONG. Aosong Eletronics Corporation LTD. **Temperature and humidity module - DHT11 Product Manual.** 2018. Disponível em: <http://akizukidenshi.com/download/ds/aosong/DHT11.pdf>. Acesso em 5 de Julho de 2018.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. **The internet of things: A survey.** Computer networks, v. 54, n. 15, p. 2787-2805, 2010. Disponível em: <https://www.researchgate.net/profile/Luigi_Atzori2/publication/222571757_The_Internet_of_Things_A_Survey/links/546b36df0cf2f5eb180914e5/The-Internet- of-Things-A-Survey.pdf>. Acesso em 17 de Março de 2017.

BIASON, Alessandro et al. **EC-CENTRIC: An energy-and context-centric perspective on IoT systems and protocol design**. IEEE Access, v. 5, p. 6894-6908, 2017. Disponível em:

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7895124>. Acesso em 2 de Maio de 2018.

BOTTA, Alessio et al. **Integration of cloud computing and internet of things: a survey.** Future Generation Computer Systems, v. 56, p. 684-700, 2016. Disponível em: <http://wpage.unina.it/valerio.persico/pubs/CloudIoT_FGCS.pdf>. Acesso em 4 de Abril de 2018.

CHEN, Yan-Da; AZHARI, Muhammad Zulfan; LEU, Jenq-Shiou. **Design and implementation of a power consumption management system for smart home over fog-cloud computing.** In: 2018 3rd International Conference on Intelligent Green Building and Smart Grid (IGBSG). IEEE, 2018. p. 1-5.

CHIANG, Mung et al. **Clarifying fog computing and networking: 10 questions and answers**. IEEE Communications Magazine, v. 55, n. 4, p. 18-20, 2017. Disponível em: <http://porto.polito.it/2679592/1/17Commag_Fog_10Questions.pdf>. Acesso em 7 de Fevereiro de 2018.

DE FARIA, Ana Cristina; SIQUEIRA, Luciene Diana; DA SILVA MARTINS, Márcia. **TI Verde: mito ou realidade na indústria digital brasileira.** Revista Electronica de Sistemas de Informaçao, v. 15, n. 1, p. 1, 2016. Disponível em: <http:// www.anpad.org.br/admin/pdf/2013_EnANPAD_ADI441.pdf>. Acesso em 5 de Abril de 2018.

DIGI. **Digi XBee Zigbee**. 2018. Disponível em: <https://www.digi.com/products/xbee-rf-solutions/2-4-ghz-modules/xbee-zigbee>. Acesso em 11 de Agosto de 2018.

DIGI. **Digi XBee Wi-Fi**. 2018. Disponível em: < https://www.digi.com/products/xbee-rf-solutions/2-4-ghz-modules/xbee-wi-fi#specifications>. Acesso em 29 de Setembro de 2018.

FILIPE FLOP. **Sensor de Chama Fogo 760 a 1100 nm**. 2018. Disponível em: <https://www.filipeflop.com/produto/sensor-de-chama-fogo/>. Acesso em 19 de Agosto de 2018.

FRIEDLI, Martin et al. **Energy efficiency of the Internet of Things. Technology and Energy Assessment Report prepared for IEA 4E EDNA.** Lucerne University of Applied Sciences, Switzerland, 2016. Disponível em: <https://www.iea-4e.org/document/384/energy-efficiency-of-the-internet-of-things-technology-and-energy-assessment-report>. Acesso em 4 de Junho de 2018.

GHAZAL, Bilal et al. **Multi control chandelier operations using XBee for home automation.** In: Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2015 Third International Conference on. IEEE, 2015. p. 107-111. Disponível em: < https://www.researchgate.net/profile/Bilal_Ghazal/publication/301403907_Multi_control_chandelier_operations_using_XBee_for_home_automation/links/5826e22508ae950ace6c51e5/Multi-control-chandelier-operations-using-XBee-for-home-automation.pdf >. Acesso em 23 de Setembro de 2018.

HANWEI. Hanwei Eletronics Co., LTD. **Technical Data MQ-2 Gas Sensor.** 2018. Disponível em: <https://raw.githubusercontent.com/SeeedDocument/Grove-Gas_Sensor-MQ2/master/res/MQ-2.pdf>. Acesso em 6 de Julho de 2018.

IEA 4E. **Annual Report 2017**. 2018. 26 p. Disponível em: <https://www.iea-4e.org/document/416/4e-2017-annual-report>. Acesso em 4 de Junho de 2018.

IORGA, Michaela et al. **Fog Computing Conceptual Model**. 2018. Disponível em: <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-325.pdf>. Acesso em 15 de Abril de 2018.

JALALI, Fatemeh et al. **Greening IoT with fog: A survey.** In: Edge Computing (EDGE), 2017 IEEE International Conference on. IEEE, 2017. p. 25-31. Disponível em: <https://www.researchgate.net/profile/Fatemeh_Jalali2/publication/318418113_Greening_IoT_with_Fog_A_Survey/links/5a1b37080f7e9be37f9bea5f/Greening-IoT-with-Fog-A-Survey.pdf>. Acesso em 7 de Fevereiro de 2018.

MANYIKA, James et al. **Unlocking the Potential of the Internet of Things.** McKinsey Global Institute, 2015. Disponível em: <http://aegex.com/images/uploads/white_papers/Unlocking_the_potential_of_the_Internet_of_Things___McKinsey__Company.pdf>. Acesso em 15 de Abril de 2018.

MARQUES, Fabrício. **O Brasil da Internet das Coisas.** Revista Pesquisa FAPESP. São Paulo, edição 259, p. 18-23. Setembro de 2017. Disponível em: <http://revistapesquisa.fapesp.br/2017/09/21/o-brasil-da-internet-das-coisas/>. Acesso em 6 de Abril de 2018.

PERERA, Charith et al. **Fog computing for sustainable smart cities: A survey.** ACM Computing Surveys (CSUR), v. 50, n. 3, p. 32, 2017. Disponível em: <https://arxiv.org/pdf/1703.07079.pdf>. Acesso em 7 de Fevereiro de 2018.

Osoyoo. **Bluetooth 4.0 BLE module Datasheet.** 2016. Disponível em: < http://osoyoo.com/wp-content/uploads/2016/10/OSOYOO-HM-10-Bluetooth-Module.pdf>. Acesso em 29 de Setembro de 2018.

SANTA CATARINA. **Instrução Normativa (IN 012/DAT/CBMSC):** Normas de segurança contra incêndios – sistema de alarme e detecção de incêndio. 28 de Março de 2014. Santa Catarina. Disponivel em: <http://www.cbm.sc.gov.br/dat/images/arquivo_pdf/IN/IN_29_06_2014/IN_12.pdf>. Acesso em 20 de Setembro de 2018.

SAMPAIO, Hugo; MOTOYAMA, Shusaburo. **"Implementation of a greenhouse monitoring system using hierarchical wireless sensor network,"** 2017 IEEE 9th Latin-American Conference on Communications (LATINCOM), Guatemala City, 2017, pp. 1-6. DOI: 10.1109/LATINCOM.2017.8240156

SINGH, Manpreet; KAUR, Ramanpreet. **Integration of IoT and Fog: Need of the Hour.** International Journal of Advanced Research in Computer Science, v. 7, n. 6, 2016. Disponível em: <http://www.ijarcs.info/index.php/Ijarcs/article/view/2741/2729>. Acesso em 4 de Abril de 2018.

STOJKOSKA, Biljana L. Risteska; TRIVODALIEV, Kire V. **A review of Internet of Things for smart home: Challenges and solutions.** Journal of Cleaner Production, v. 140, p. 1454-1464, 2017. Disponível em: <https://www.researchgate.net/profile/Biljana_Risteska_Stojkoska/publication/308975029_A_review_of_Internet_of_Things_for_smart_home_Challenges_and_solutions/links/5af533a34585157136ca43a3/A-review-of-Internet-of-Things-for-smart-home-Challenges-and-solutions.pdf>. Acesso em 23 de Setembro de 2018.

VARGHESE, Blesson et al. **Feasibility of Fog Computing.** arXiv preprint arXiv: 1701.05451, 2017. Disponível em: <https://arxiv.org/pdf/1701.05451.pdf>. Acesso em 4 de Abril de 2018.

ZigBee Alliance. **ZigBee-PRO Stack Profile: Platform restrictions for compliant platform testing and interoperability.** 2014. Disponível em: <https://www.zigbee.org/download/standard-zigbee-pro-specification/>. Acesso em 4 de Agosto de 2016.

ZIMMERMANN, André et al. **Arquitetura para ganho de eficiência energética em redes de sensores sem fios de próxima geração.** 2008. Disponível em: <https://repositorio.ufsc.br/handle/123456789/91383>. Acesso em 30 de Maio de 2018.

Apêndices

Apêndice A – Algoritmo em linguagem C para captura de dados pelo nó sensor

```
int umidade = 0;
int temperatura = 0;
int sensorChamas = 0;
int pinSensorChama = A3;
int pinSensorGas = A0;
int ValDesarm = 0; //Variável para selecionar a quantidade de Gás/Fumaça detectada
int valorSensorFumaca = 0;
int temp = 0;
int ciclo = 0;
boolean incendio = false;
int Talarme = 0;

unsigned long intervaloAr = 2000;
unsigned long intervaloChamas = 100;
unsigned long intervaloGas = 1000;
unsigned long inicioAr = 0;
unsigned long inicioChamas = 0;
unsigned long inicioGas = 0;

#include "DHT.h"
#include <Adafruit_Sensor.h>
#include <SoftwareSerial.h>
#include <Thread.h>
#include <ThreadController.h>

ThreadController iot;
ThreadController sensores;
ThreadController sensores2;
Thread Thread_Temperatura;
Thread Thread_Gas;
Thread Thread_Chamas;
Thread Alarme;
Thread Antena;

#define DHTPIN A1
#define DHTTYPE DHT11
DHT dht (DHTPIN, DHTTYPE);

SoftwareSerial zigbee(2, 3);
```

```
void setup() {

Serial.begin(115200);

zigbee.begin(115200);

Serial.print("b");

zigbee.print("b");

dht.begin();



Antena.setInterval(0);

Antena.onRun(checarAntena);



Thread_Temperatura.setInterval(0);

Thread_Temperatura.onRun(capturarTem

peraturaUmidade);



Thread_Gas.setInterval(0);

Thread_Gas.onRun(capturarGasFumaca);



Thread_Chamas.setInterval(0);

Thread_Chamas.onRun(capturarChamas);



Alarme.setInterval(Talarme);

Alarme.onRun(alarme);

Alarme.onRun(enviar);
```

```
iot.add(&Antena);
sensores.add(&Thread_Temperatura);
sensores.add(&Thread_Gas);
sensores.add(&Thread_Chamas);

pinMode(7, OUTPUT); //pinos de energia
//dos sensores, para ligar e desligar --
//sensor temp dht11
pinMode(11, OUTPUT); // sensor gas
pinMode(10, OUTPUT); // sensor chamas
inicioAr = 0;
inicioChamas = inicioAr + (intervaloAr -
intervaloChamas);
inicioGas = inicioAr + (intervaloAr -
intervaloGas);

}


void loop() {

  iot.run();
 // imprimeContadores();
  if (incendio == false){
    sensores.run();
    enviar();
  }
  else if (incendio == true){
    Alarme.run();
    enviar();
  }
}
```

```
void alarme(){
   digitalWrite(7, HIGH);
    digitalWrite(10, HIGH);
    digitalWrite(11, HIGH);
    umidade = dht.readHumidity();
    temperatura = dht.readTemperature();
    valorSensorFumaca = analogRead(pinSensorGas); //Faz a leitura da entrada do
//sensor de fumaça
    valorSensorFumaca = map(valorSensorFumaca, 0, 1023, 0, 100); //Faz a conversão
//da variável para porcentagem
    sensorChamas = analogRead(pinSensorChama);
    if (sensorChamas < 1000) {
      sensorChamas = 0;
    } else {
      sensorChamas = 1;
    }
}


void enviar() {
  enviarTela();
  enviarAntena();
}

void imprimeContadores() {

  Serial.println(inicioAr);
  Serial.println(inicioGas);
  Serial.println(inicioChamas);

}

void checarAntena() { //Antena modo AT
  if (zigbee.available() > 0){
    switch (zigbee.read()) {
     case 0x4C:
     incendio = true;
     break;

     case 0x44:
     incendio = false;
     break;
    }
  }
```

```
void capturarTemperaturaUmidade () {
  if (millis() > inicioAr) {
    digitalWrite(7, HIGH);
    Serial.println("Ar Ligado");

  if (millis() > inicioAr + intervaloAr)
  {
    inicioAr = millis();
    Serial.println("Ar Desligado");
    umidade = dht.readHumidity();
    temperatura = dht.readTemperature();
    digitalWrite(7, LOW);
  }
  }
}

void capturarGasFumaca () {
  if (millis() > inicioGas) {
    digitalWrite(11, HIGH);
    Serial.print("Gas Ligado");

  if (millis() > inicioGas + intervaloGas){
        inicioGas = inicioAr + (intervaloAr - intervaloGas);
    valorSensorFumaca = analogRead(pinSensorGas); //Faz a leitura da entrada do
sensor de fumaça
    digitalWrite(11, LOW);
    valorSensorFumaca = map(valorSensorFumaca, 0, 1023, 0, 100); //Faz a conversão
da variável para porcentagem
    Serial.print("Gas desligado");
  }}
}

void capturarChamas() {
    if (millis() > inicioChamas) {
      digitalWrite(10, HIGH);
      Serial.print("Chamas Ligado");

    if (millis() > inicioChamas + intervaloChamas){
      inicioChamas = inicioAr + (intervaloAr - intervaloChamas);
      sensorChamas = analogRead(pinSensorChama);
      digitalWrite(10, LOW);
      if (sensorChamas < 1000) {
        sensorChamas = 0;
      } else {
        sensorChamas = 1;
      }
      Serial.print("Chamas Desligado");
    }
    }
}
```

```
void enviarTela() {
  Serial.print(umidade);
  Serial.print(temperatura);
  Serial.print(sensorChamas);
  Serial.println(valorSensorFumaca);
}

void enviarAntena() {
  zigbee.print(umidade);
  zigbee.print(temperatura);
  zigbee.print(sensorChamas);
  zigbee.print(valorSensorFumaca);
}
```

Algoritmo 1 - Algoritmo em linguagem C para captura de dados pelo nó sensor

Apêndice B – Algoritmo em linguagem C para recebimento e envio de dados no nó *Mist*

```
byte pula = 0;
int x = 0;
int ar = 0;
int ar2 = 0;
int temp = 0;
int temp2 = 0;
int aux = 0;
int aux2 = 0;
int pacoteOrigem[20];
int pacote [4];
int comando = 0;
int i = 0;
int gas =0;
int gas2 =0;
int gasTotal=0;
int chama=0;
int inicio = 0;
int fim = 0;
#include <SoftwareSerial.h>
#include <Thread.h>
#include <ThreadController.h>
```

```
ThreadController fog;
Thread Antena;
Thread serial;

SoftwareSerial zigbee(2,3);

void setup() {
  Serial.begin(115200);
  Serial.print("b");
  delay(100);
  zigbee.begin(115200);
  zigbee.print("b");
  Antena.setInterval(1);
  Antena.onRun(checarAntena);
  serial.setInterval(1);
  serial.onRun(checarSerial);
  fog.add(&Antena);
  fog.add(&serial);
  pinMode(8,OUTPUT);
}
```

```
void loop() {
 fog.run();
}

void checarAntena(){
  if (zigbee.available() >= 19)
  {
   //Serial.print("ok");
    if (zigbee.read() == 0x7E)
    {
      for(i = 0; i < 20; i++)
      {
        pacoteOrigem[i] = zigbee.read();
      // Serial.println(pacoteOrigem[i], HEX);
      }
     enviaFog();
    }
  }
}

void checarSerial(){
  if (Serial.available() > 0)
  {
    if (Serial.read() == 0x11)
    {
      comando = Serial.read();
      switch (comando) {
        case 0x61: //ativa ou desativa alarme de incendio
          for(i=0 ; i<3; i++) { //2 bytes endereço destino, 1 byte valor a
            pacote[i] = Serial.read();
          }
          enviarComandoA();
          break;
        case 0x62: /Altera frequencia de sleep
          for(i=0 ; i<3; i++) {
            pacote[i] = Serial.read();
          }
          enviarComandoB();
          break;
      }
     }
   }
}
```

```
void enviarComandoA(){ //envio de pacote API


  if (pacote[2] == 'D'){
  byte pacoteD[] = {0x7E, 0x00, 0x0F, 0x10, 0x01, 0x00, 0x13, 0xA2, 0x00, 0x40,
                    0xD9, 0xAA, 0xA8, 0xFF, 0xFE, 0x00, 0x00, 0x44, 0x8D};
  zigbee.write(pacoteD, sizeof(pacoteD));
  }

  if (pacote[2] == 'L'){
  byte pacoteL[] = {0x7E, 0x00, 0x0F, 0x10, 0x01, 0x00, 0x13, 0xA2, 0x00, 0x40,
                    0xD9, 0xAA, 0xA8, 0xFF, 0xFE, 0x00, 0x00, 0x4C, 0x85};
  zigbee.write(pacoteL, sizeof(pacoteL));
  }
}
void enviarComandoB(){

  byte pacoteB[] = {0x7E, 0x00, 0x11, 0x10, 0x01, 0x00, 0x13, 0xA2, 0x00, 0x40,
                    0xD9, 0xAA, 0xA8, 0xFF, 0xFE, 0x00, 0x00, 0x46, 0x33, 0x58};
  zigbee.write(pacoteB, sizeof(pacoteB));
}
void enviaFog() {
  if (pacoteOrigem[10] == 0xA8) {
     // Serial.print("ok2");
      ar = pacoteOrigem[16];
      ar2 = pacoteOrigem[17];
      temp = pacoteOrigem[14];
      temp2 = pacoteOrigem[15];
      ar = (ar - 48) * 10;
      ar2 = ar2 - 48;
      ar = ar + ar2;
      temp = (temp - 48) * 10;
      temp2 = temp2 - 48;
      temp = temp + temp2;

      gas = pacoteOrigem[18];
      gas2 = pacoteOrigem[19];
      gas = (gas - 48) * 10;
      gas2 = gas2 - 48;
      gas = gas + gas2;

      chama = pacoteOrigem[20];

      imprimir();

    }
  }
```

```
void imprimir () {
  Serial.write(0x11);
  Serial.write(pacoteOrigem[9]);
  Serial.write(pacoteOrigem[10]);
  Serial.print(aux);
  Serial.print(aux2);
  if (gasTotal < 10){
  Serial.write(0x30);
  }
  Serial.print(gasTotal);
  Serial.print(chama);

}
```

Algoritmo 2 - Algoritmo em linguagem C para recebimento e envio de dados no nó *Mist*

Apêndice C – Artigo no formato SBC

Gerenciamento autonômico de dispositivo de Internet of Things com restrição de energia utilizando arquitetura de Fog Computing

Ana Luiza Córdova de Jesus

Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brasil
Departamento de Informática e Estatística

analuizacordova@gmail.com

Abstract: With the rise of Internet of Things paradigm in recent years, there has been also a growth in interest in Fog Computing. One way to integrate both technologies to benefit its users is to apply those technologies in Smart Home scenarios, and with that make use of common devices in an automated way (even keeping in mind having a smaller energy consumption). With that in mind, this work proposes to create a prototype of a smart fire alarm, using concepts of Fog Computing and Internet of Things, as well as measuring the lifetime span of the prototype. Finally, comparisons of lifetime span are performed, according with the network protocol used in the prototype.

Resumo: Com o crescimento da *Internet of Things* nos últimos anos, houve também o crescimento da *Fog Computing*. Uma das maneiras de utilizar as duas tecnologias de modo a beneficiar os usuários é aplica-las em casas inteligentes, de modo a tornar o uso de dispositivos comuns de forma automatizada, e muitas vezes trazendo uma economia de energia. Com isso em mente, é proposto a criação de um protótipo de alarme de incêndio inteligente, utilizando conceitos de *Internet of things* e *Fog* computing, de modo a medir o tempo de vida útil desse protótipo, de acordo com o gasto de energia do mesmo. Finalmente, são realizadas comparações de cálculos de tempo de vida, variando o protocolo de rede utilizado (ZigBee, Wi-Fi e Bluetooth).

1. Introdução

É inegável o crescimento do uso de dispositivos inteligentes na sociedade atual. Uma das formas de aplicar essa tecnologia e fazer uso de dispositivos inteligentes de maneira a garantir a segurança de seus usuários, como por exemplo, utilizar conceitos e tecnologia de casas inteligentes e *Internet of Things* para situações de emergência, como por exemplo, um incêndio. Um alarme de incêndio inteligente conectado à rede de uma residência, que utilize formas de captura e envio de dados para um usuário e que ainda assim tenha uma preocupação com o gasto de energia desse dispositivo é algo que une a crescente onda de *Internet of things* e *Fog computing*. Assim, foi pensado em uma solução para evitar esse tipo de acidente, utilizando *Internet of Things* e *Fog computing* de modo a ser possível realizar o monitoramento e envio de dados para o usuário em tempo real.

2. Conceitos Fundamentais

2.1 Fog Computing

Fog Computing (ou Computação na Neblina, em português), pode ser considerada como uma extensão da computação em nuvem, cujo foco é em aplicações que precisam de poucos atrasos, e análises em tempo real (Singh; Kaur, 2016). A definição mais formal define *Fog Computing* como “um modelo em camadas que permite o acesso onipresente a recursos computacionais escaláveis” (Iorga et al, 2018). Este modelo é composto de nodos que podem ser físicos ou virtuais, e que se encontram localizados entre dispositivos considerados inteligentes e serviços centralizados de computação na nuvem (Iorga et al, 2018).

Seu nome vem de uma das principais diferenças com a computação em nuvem - a idéia que a neblina se encontra mais próxima do chão do que a nuvem, e isso se reflete em uma maior proximidade do usuário com o servidor (Botta et al, 2016).

Até 2020 estima-se que entre 20 e 50 bilhões de dispositivos novos serão conectados à Internet, gerando um aumento de 3 trilhões de dólares à economia mundial (Varghese et al, 2017). Dessa forma, a *Fog Computing* começa a se estabelecer como uma ótima alternativa à computação na nuvem, principalmente no que se diz em respeito a *Internet of Things*.

2.2 Internet of Things

Internet of Things (IoT, Internet das Coisas em português) é considerada como um paradigma relativamente novo na área de telecomunicações modernas (Atzori; Iera; Morabito,

2010). A ideia básica, porém, pode ser definida como uma força omnipresente onde objetos e tecnologias como sensores, celulares, etiquetas com identificação por radiofrequência, entre outros, interagem e trabalham entre seus vizinhos com a finalidade de atingir um objetivo comum (Atzori; Iera; Morabito, 2010).

Algumas aplicações em IoT utilizam direção assistida (carros inteligentes), mapas online com realidade aumentada, casas inteligentes (como possibilidade de monitorar o interruptor de lâmpadas, sistema de aquecimento, sistemas de alarme, etc), uso em redes sociais para atualização automática, e diversas outras (Atzori; Iera; Morabito, 2010).

O potencial financeiro da difusão da *Internet of Things* também é algo que deve ser considerado, pois estima-se que até o ano 2025 a mesma deve ter um impacto de $11 trilhões ao ano - o equivalente a 11% da economia mundial (Manyika et al, 2015). As áreas da economia mais afetadas, e consequentemente, mais beneficiadas com o uso amplo e difundido da IoT são fábricas industriais, cidades inteligentes, monitoramento de doenças em humanos, direção autônoma de veículos, entre outros (Manyika et al, 2015).

3. Aspectos Relevantes

Dentro dos estudos selecionados como base referencial desse trabalho, foi possível observar alguns dos aspectos e problemas existentes quando se diz respeito a Fog Computing e Internet of Things.

3.1 Consumo e economia de energia

Economia de energia de dispositivos de IoT, principalmente quando integrando *Fog Computing*, pode ser visto de diversos ângulos. Primeiramente, deve-se ter em mente que nem todas aplicações tiram vantagem do uso de *Fog* - aplicações e serviços que necessitem de muitos *downloads*, *updates* ou necessitem de computações que são consideradas muito pesadas irão economizar mais energia quando enviadas à *Cloud*, sem tentar realizar essas tarefas na borda da rede antes (Jalali et al, 2017). Dessa forma, é melhor focar em aplicações de tempo real e baixa latência quando for decidido tirar vantagem das características diferenciadas do *Fog.*

Uma questão importante quando se fala de economia de energia utilizando Internet das Coisas, é o fato que muitos dispositivos de IoT dependem de baterias para funcionar e também se encontram conectados a redes de comunicação sem fio. Para usuários desses dispositivos,

uma longa vida útil de um dispositivo é considerada importante; ou seja, o dispositivo dependente da bateria deve conseguir durar muitos anos (Friedli et al, 2016).

3.2 Protocolos de rede

Protocolos de rede, embora não sejam pensados como a principal fonte de gasto de energia em dispositivos de *Internet of Things*, podem desempenhar um papel importante na identificação de fontes de consumo de energia em um dispositivo, assim como na ineficiência da energia do mesmo (Biason, 2017), de acordo com resultados de cálculos de estimativa de energia.

O uso de protocolos de rede como Bluetooth e ZigBee podem ser boas opções para economia de energia em *Fog Computing* (Jalali et al, 2017). Deve-se ter em mente, porém, que diferentes dispositivos de IoT podem utilizar diferentes tecnologias a seu favor para uma maior economia de energia. Por exemplo, ZigBee pode ser utilizado em sensores, atuadores e mesmo lâmpadas LED smart (embora necessite de um *gateway*), mas não é considerado apropriado para uso em gateways e câmeras de monitoramento (Friedli et al, 2016).

Mesmo o uso de protocolos de roteamento como IPv4 ou IPv6 podem apresentar um gasto de energia quando conectando dispositivos de IoT com a Internet (Zimmerman et al, 2008).

3.3 Automação Residencial

O uso de automação residencial (ou casas inteligentes) pode ser visto como uma das áreas em que a Internet das Coisas pode ser aplicada, visto que o uso de dispositivos inteligentes do cotidiano - tal como lâmpadas, portas e janelas - pode ser caracterizado como utilização de dispositivos de Internet das Coisas (Stojkoska; Trivodaliev, 2017).

Atualmente, existe um terreno muito fértil quando se fala de automação residencial. Isso significa tornar o uso de objetos comuns do cotidiano mais inteligente, com o uso de tecnologias de comunicação, para uma melhora na qualidade de vida dos habitantes de uma residência. Ou seja, automação residencial utiliza tecnologias como conexão com rede (usando Bluetooth, Wi-Fi, e outros protocolos de rede) para conectar dispositivos de uso doméstico tal como luzes, portas, cortinas, sistema de ventilação, entre outros, e realizar o uso dos mesmos de forma mais inteligente. O exemplo mais simples e um dos mais utilizados é o uso de lâmpadas inteligentes que podem ser ligadas, desligadas e ter sua luminosidade controlada através do uso de aplicativos em dispositivos móveis como o telefone celular (Ghazal et al, 2015).

O uso da automação residencial pode ser facilmente integrado com *Fog Computing*, devido a integração de *Fog* com IoT já discutido; no caso da automação residencial, os mesmos paradigmas, vantagens e desafios ainda se aplicam.

Com o crescimento de dispositivos IoT, e consequentemente a existência de mais dispositivos domésticos que podem ser considerados inteligentes, é necessário pensar em casos de economia de energia com esse tipo de dispositivo (Stojkoska; Trivodaliev, 2017).

4. Desenvolvimento do Sistema

A partir desse capítulo é delineado o sistema desenvolvido neste trabalho, que inclui o projeto de *hardware* do protótipo de alarme de incêndio, programa de *software* para a captura de dados e protocolo de envio desses dados capturados para a Fog.

Tendo em mente a necessidade da longevidade de baterias em dispositivos IoT, assim como diminuir o consumo de energia dos mesmos, é proposto o desenvolvimento de um protótipo de dispositivo IoT protótipo com utilização de placa microcontroladora, antena e sensores, de forma a considerar as limitações e consumo energético do hardware. Fazendo uso da *Internet of Things* e do conceito de automação residencial, o protótipo é de um alarme de incêndio inteligente para detecção de valores de umidade e temperatura, gás e fumaça e presença de chamas, cujos dados capturados são enviados para um nó coordenador na *Fog* para pré- processamento dos dados.

Com isso, é possível realizar uma análise do gasto de energia do protótipo, estimando o tempo de vida do nó quando utilizando baterias para funcionamento do mesmo.

4.1 Projeto de nós

O projeto dos nós é dividido em duas partes, sendo que a primeira diz respeito a parte de hardware do alarme de incêndio, e a segunda parte é relativa a rádio, de modo a transmitir os dados capturados pela parte de hardware em uma rede sem fio.

Para o desenvolvimento dos nós do protótipo foi escolhido utilizar peças com alta disponibilidade no mercado, e que também possuem preços acessíveis. Dessa forma, a placa de prototipação selecionada foi o Arduino Uno com microcontrolador ATmega328P (Arduino, 2018). A antena escolhida para uso no protótipo foi a Xbee ZigBee, que é fabricada de acordo com as especificações do protocolo ZigBee (Digi, 2018). Todos os nós da rede, sendo eles nós sensores, coordenador e roteador, utilizaram uma placa Arduino e uma antena Xbee ZigBee.

4.1.1 Projeto da parte de hardware

O Arduino Uno (Arduino, 2018) foi selecionado como *hardware* principal do protótipo. O Arduino Uno é uma placa de microcontrolador baseada no ATmega328P, possuindo 14 portas digitais de entrada e saída, 6 portas de entrada analógica, botão de reset da placa, conexão via USB, e freqüência de clock de 16 MHz.

Três sensores foram utilizados no protótipo, um sensor de gás e fumaça (MQ-2), um sensor de chama, e um sensor de temperatura e umidade do ar (DHT11).

O dispositivo IoT também possui a capacidade de ser atuador; assim, caso sejam detectados níveis/valores acima de um limite pré-estabelecido, serão acionados avisos sonoros (com o uso de um buzzer), um aviso visual (com o uso de um LED), e será enviada uma mensagem para a *Fog* indicando perigo.

4.1.2 Projeto da parte de rádio

O protocolo ZigBee é um protocolo baseado no padrão IEEE 802.15.4, de baixa velocidade de transmissão e baixo consumo de energia, principalmente quando comparado com outros protocolos amplamente utilizados, como Wi-Fi e Bluetooth (Chen; Azhari; Leu, 2018). Dessa forma o mesmo foi escolhido para ser utilizado no desenvolvimento do protótipo IoT.

Para o componente de rádio propriamente dito, foi escolhido utilizar a antena Xbee ZigBee, especialmente devido a compatibilidade de uso com o protocolo ZigBee, além do protocolo Bluetooth. A antena Xbee Zigbee é fabricada pela empresa Digi, de acordo com as especificações do protocolo ZigBee. O componente pode ter até 14 nós filhos, com alcance máximo de 60 metros em ambientes urbanos (fechados). O Xbee ZigBee possui também freqüência de 2.4 GHz, com velocidade de transmissão de 250 Kbps (Digi, 2018).

4.2 Projeto de Nó Coordenador e Nó Roteador

Os nós que servem como coordenador e roteador no protótipo possuem apenas ligações entre a antena Xbee e a placa Arduino Uno. Essas ligações são realizadas entre as portas de transmissão e recebimento no Arduino e na antena (TX e RX, respectivamente, em ambos os componentes), assim como ligações de energia para o funcionamento dos componentes - conexão de 3.3 volts que é necessária para ligar a antena Xbee, assim como ligação de ground (GND) entre a antena e o Arduino. Essas conexões físicas são exemplificadas na Figura 1.

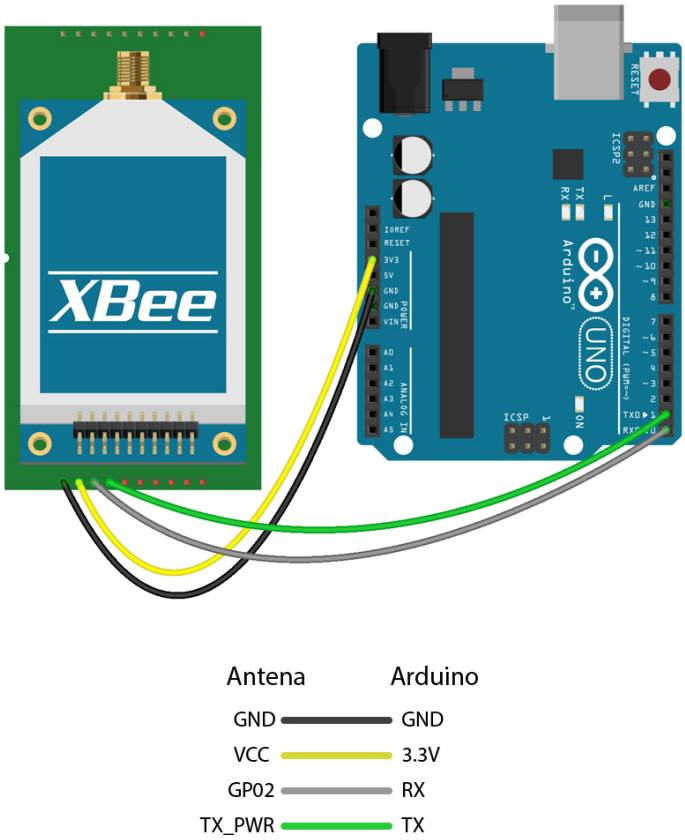


Figura 1 – Conexão física entre o Arduino e a antena XBee ZigBee

4.3 Montagem dos pacotes

Em nós sensores com a antena XBee no modo AT, o envio de dados capturados para o coordenador é realizado tal que, quando um nó sensor captura um valor, esse é armazenado na variável “x” no microcontrolador da placa do Arduino. Depois disso, o microcontrolador executa o comando “Serial.print(x);”, de forma a enviar a variável “x” da porta TX do Arduino para a porta RX da antena. A antena recebe esse valor, como um sinal imprimível, conforme a tabela ASCII (*American Standard Code for Information Interchange*), e procede a encapsular automaticamente o valor capturado pelo sensor e então envia-lo para o nó coordenador (Sampaio e Motoyama, 2017).

Os pacotes enviados de um nó sensor para o coordenador possuem tamanhos variáveis de acordo com a quantidade de dados que devem ser enviados; esse tipo de pacote é chamado de “pacote de requisição de transmissão”.

Para exemplificar o modo de transmissão de dados, pode-se pensar em uma situação hipotética onde, em um nó sensor, foram capturados os valores 25 graus Celsius e 74% de umidade no sensor DHT11, valor 99% de presença de gás e/ou fumaça no sensor MQ-2 e o

valor 1 (indicando fogo) no sensor de chama. Esses valores serão armazenados em variáveis separadas “temperatura”, “umidade”, “gasTotal” e “chama”, identificados pelos comandos do Quadro 1, onde é possível enviar para a porta TX do Arduino os valores capturados, que então serão enviados a porta RX da antena, onde será realizada a leitura dos dados.

Quadro 1 – Comandos para envio dos dados capturados pelo nó sensor

```
void enviar() {
        Serial.print(umidade);
        Serial.print(temperatura);
        Serial.print(sensorReading);
        Serial.print(valorSensorFumaca);
}
```

A antena por sua vez, transformará os valores decimais capturados em valores hexadecimais, um caractere por vez, separadamente. Ou seja, o valor decimal 25 será transformado para os valores hexadecimais 0x32 e 0x35, onde cada caractere ocupa 1 byte de tamanho no pacote de transmissão. O mesmo acontece com o restante dos valores.

Após a transformação dos valores capturados, eles serão encapsulados de maneira automática em um pacote, e finalmente enviados para o nó coordenador. O Quadro 2 mostra em detalhes um exemplo de um pacote criado com o comando enviarPacote(), com tamanho de 25 bytes.

Quadro 2 – Detalhes de um pacote de requisição de transmissão.

Byte	Descrição	Valor Hexadecimal	Significado
1	Delimitador inicial	7E	
2 e 3	Tamanho do pacote	00 15	21 bytes de tamanho
4	Tipo do frame	10	Requisição de transmissão
5	ID do frame	01	Identificação do frame
6 a 13	Endereço de destino de 64-bit	00 13 A2 00 40 D9 AA 6B	Endereço do nó coordenador
14 e 15	Endereço de destino de 16-bit	FF FE	Envio para todos os roteadores
16	Raio de broadcast	00	Pulos máximos de broadcast
17	Opções	00	
18 a 24	Dados	32 35 37 34 39 39 31	Dados capturados pelo nó sensor, com valores da tabela ASCII
25	Checksum	19	

Quando o pacote é recebido no nó coordenador, a antena processa o pacote e na porta RX é feita a substituição dos bytes de endereço de destino por bytes indicando o nó que enviou o pacote, e envia em sua porta RX para porta TX do Arduino (Sampaio e Motoyama, 2017).

O envio dos dados entre o nó sensor e o nó coordenador pode ser exemplificado pela Figura 2, onde é exibido o envio de um pacote de transmissão de dados capturados pelo nó sensor. Quando o nó coordenador recebe esse pacote, ele realiza o processamento do mesmo, assim como envio para a porta serial. A porta serial deve processor e extrair as informações capturadas pelo sensor, podendo estar conectada a um computador ou a uma placa Arduino (Sampaio e Motoyama, 2017).

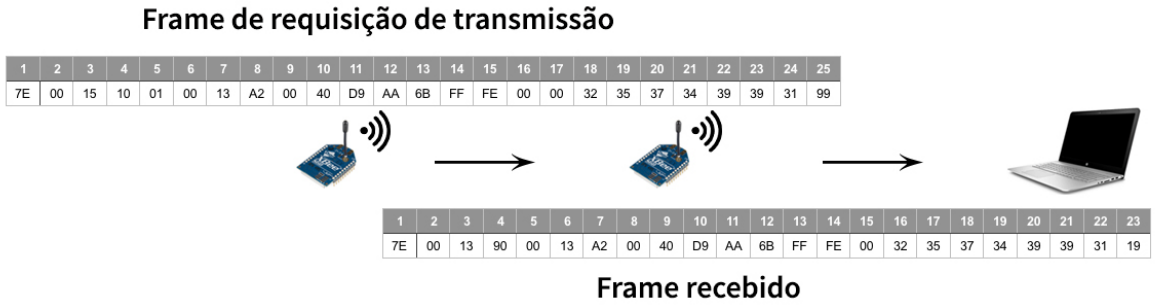


Figura 2 - Exemplificação de envio de dados do nó sensor para o nó coordenador

É necessário notar que, ao ser recebido pelo nó coordenador, o pacote "perde" dois bytes - referentes aos campos Raio de Broadcast e ID do frame – pois os mesmos são utilizados apenas por pacotes de envio de transmissão, não em pacotes recebidos. O Quadro 3 mostra os detalhes de um pacote de recebimento.

Quadro 3 – Detalhes de um pacote de recebimento

Byte	Descrição	Valor Hexadecimal	Significado
1	Delimitador inicial	7E	
2 e 3	Tamanho do pacote	00 13	19 bytes de tamanho
4	Tipo do frame	90	Frame recebido
5 a 12	Endereço de origem de 64-bit	00 13 A2 00 40 D9 AA 6B	Endereço do nó sensor
13 e 14	Endereço de origem de 16-bit	FF FE	Endereço local do nó sensor
15	Opções	00	
16 a 22	Dados	32 35 37 34 39 39 31	Dados capturados pelo nó sensor
23	Checksum	19	

4.4 Nó Sensor

Foram selecionados três sensores para uso no protótipo: de temperatura e umidade (capturados pelo sensor DHT11), de chamas (capturada pelo sensor de chamas), e de fumaça e

gases (capturados pelo sensor MQ-2). Além disso, foram utilizados alertas físicos no evento de disparo do alarme, na forma de um LED e um buzzer passivo.

Pensando na economia de energia de modo geral para o protótipo, ao invés de conectar os sensores diretamente no 5V do Arduino, os sensores foram conectados a portas digitais. Dessa forma, os sensores gastam energia apenas quando as portas digitais estão ligadas, sendo fácil programar a ligação e o desligamento das portas digitais.

Ao utilizar portas digitais, existe a necessidade de realizar uma pequena alteração relativo ao sensor de gás MQ-2, já que o mesmo gasta 160mA (Hanwei, 2018), porém uma placa Arduino UNO aceita apenas 20mA por porta digital. Assim, não há como energizar o sensor diretamente na porta digital, faz-se com que seja necessário o uso de um transistor NPN BC327 para o uso correto do sensor MQ-2.

Esse transistor faz a mesma função de uma porta digital, ou seja, de controle da passagem de energia. Com o uso do transistor, é possível fazer o controle da saída de voltagem (de 5V) do Arduino, através de uma porta digital.

Assim, todas as conexões físicas do nó sensor – incluindo os três sensores, antena XBee, Arduino UNO e avisos físicos, sem encontram exemplificadas na Figura 3.

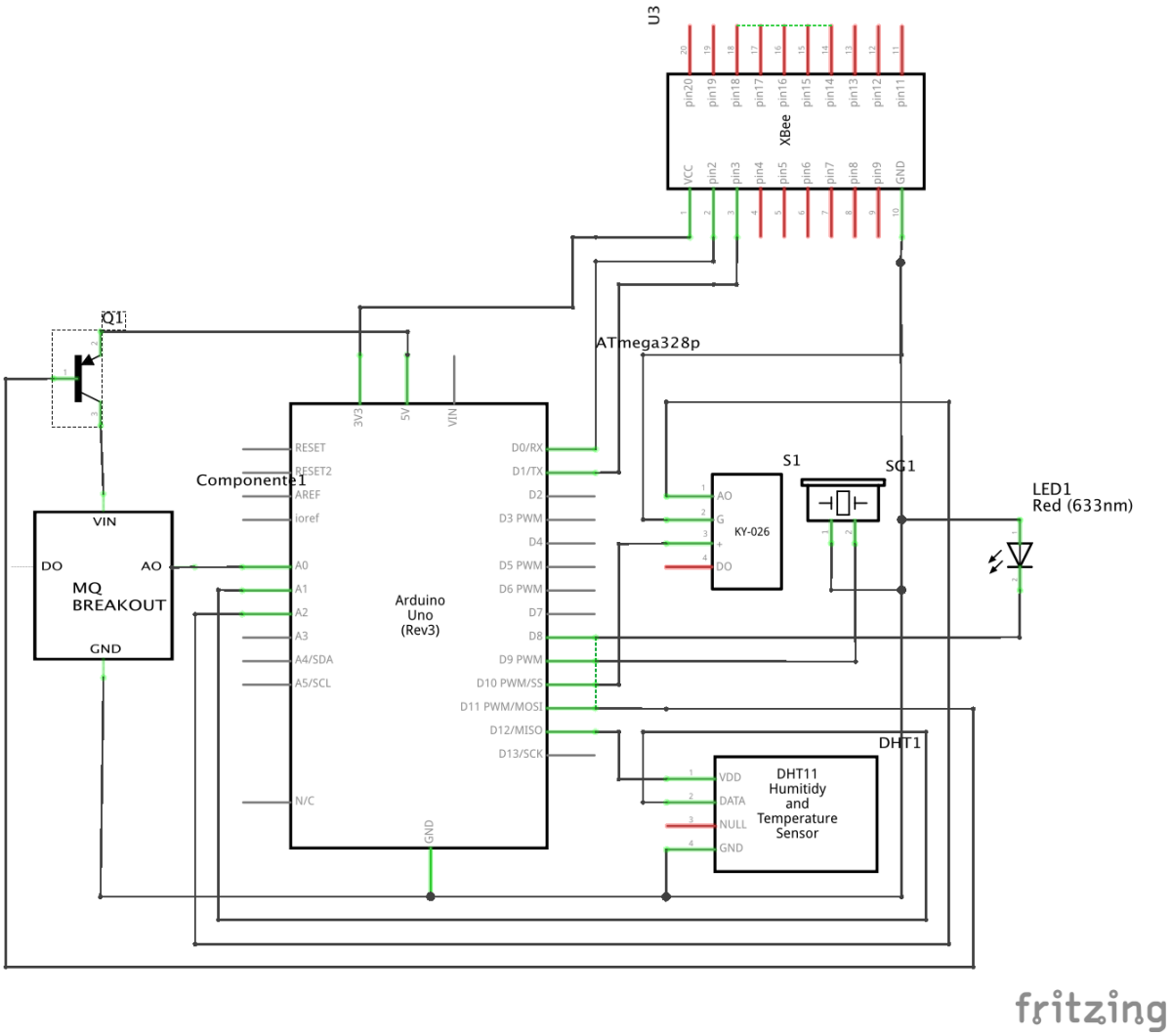


Figura 3 – Conexão física do nó IoT usando portas digitais

5. Gerenciamento de dispositivo IoT com Fog

A arquitetura utilizada no protótipo, envolvendo o dispositivo IoT, o nó *Mist* e a *Fog* é mostrada, em seu total, na Figura 4. Embora na figura esteja sendo considerada uma arquitetura completa com quatro níveis (*Cloud*, *Fog*, *Mist* e nó IoT), o trabalho desenvolveu e focou apenas na troca de mensagens entre o nó *Mist* e o nó IoT.

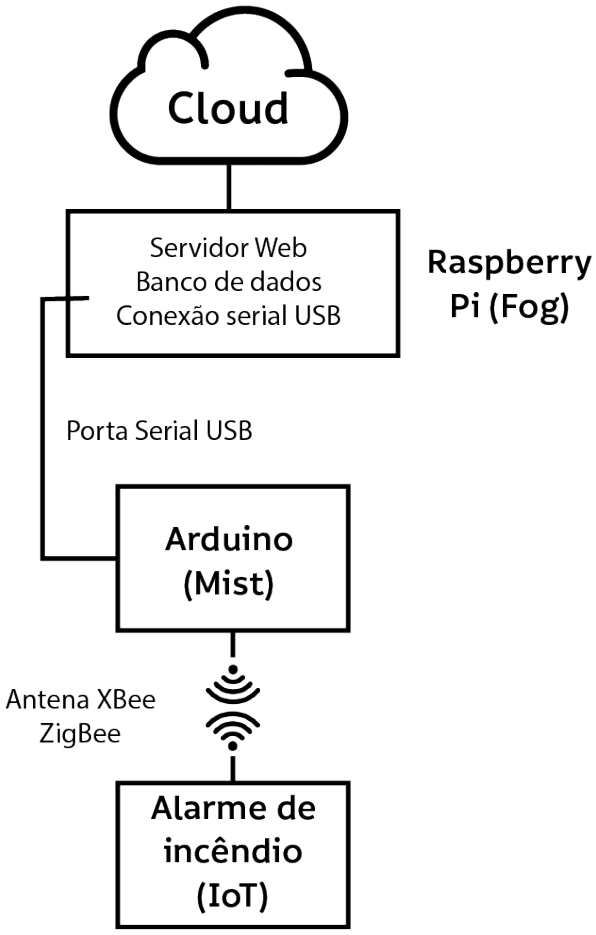


Figura 4 - Arquitetura utilizada no desenvolvimento deste trabalho.

5.1 Exibição dos possíveis estados do protótipo para um usuário

A condição de perigo é indicada com a combinação/cruzamento dos valores dos sensores do ar e gás, pois caso haja valor alto de fumaça - porém sem aumento de temperatura - pode indicar um fósforo que foi aceso, sem o acontecimento de um princípio de incêndio. Os possíveis estados do protótipo se encontram no Quadro 4.

Quadro 4 – Possíveis estados do protótipo IoT

Estados possíveis (Temperatura e umidade, Gás e fumaça, chamas)	Temperatura e Umidade	Gás e fumaça	Chamas	Condição Definida
(0, 0, 0)	Baixa	Baixo	Não	Condição normal
(1, 0, 0)	Alta	Baixo	Não	Abertura da porta do fogão, saindo ar quente
(0, 1, 0)	Baixa	Alto	Não	Vazamento de gás
(1, 0, 1)	Alta	Baixo	Sim	Erro de sensor (falso positivo)
(0, 0, 1)	Baixa	Baixo	Sim	Acendimento de fósforo
(0, 1, 1)	Baixa	Alto	Sim	Erro de sensor (falso positivo)
(1, 1, 0)	Alta	Alto	Não	Alimento queimando dentro do fogão
(1, 1, 1)	Alta	Alto	Sim	Incêndio

Os valores definidos como limite para cada um dos sensores foram de 30ºC de temperatura, 4% de gás e fumaça e sem presença de chamas. Quaisquer valores acima desses são considerados valores “altos”, e apresentam diferentes cenários para cada um dos possíveis estados do protótipo.

Esses possíveis estados, e consequentemente as situações que eles representam, poderão ser acompanhados através do uso de um sistema desenvolvido pelo LRG (Laboratório de Redes e Gerência) da UFSC (Universidade Federal de Santa Catarina). Dessa forma também é possível realizar ações como ligar ou desligar o alarme.

5.2 Comunicação entre Mist e Fog

O conceito de *Fog Computing* é por natureza considerado mais leve e mais próximo do usuário do que a *Cloud*. Ainda assim, é possível usar um tipo de estrutura subsidiária da *Fog* – a *Mist Computing*. *Mist Computing* é definido como uma camada mais leve da *Fog*, com nós especializados, e principalmente com baixo poder computacional, mas ainda residindo dentro da rede de *Fog*. Esses nós, denominados “nós *Mist*” usam microcontroladores que alimentam nós da *Fog* (Iorga et al, 2018). No caso dos nós apresentados aqui, o nó coordenador pode ser considerado como nó *Mist*, devido ao baixo poder computacional do Arduino. Quanto ao nó sensor, esse é considerado apenas como um nó IoT.

Depois da captura de dados pelo nó IoT, esses dados são enviados para o nó *Mist* através da antena XBee. O nó *Mist*, por sua vez, encontra-se conectado a um Raspberry Pi, e realiza um pré-processamento dos dados (envio apenas do endereço do nó IoT e dados capturados, ao invés de um pacote inteiro padrão) antes de enviá- los ao Raspberry Pi (sendo ele o nó mais forte da *Fog*).

Para o envio e recebimento de pacotes entre a *Fog*, *Mist* e o nó IoT foi criado um protocolo novo especificamente para atender as necessidades dos dispositivos envolvidos no trabalho aqui desenvolvido. Esse protocolo foi baseado no protocolo ZigBee, embora ele esteja estruturado de forma que também seja possível realizar uma "tradução" de e para outros protocolos, como o Wi-Fi e o Bluetooth. A Figura 5 exemplifica este pacote reduzido, que é enviado ao Raspberry Pi pelo nó *Mist*.

	Byte ID	Nó Origem		Umidade		Temperatura		Gás		Chamas
Byte	1	2	3	4	5	6	7	8	9	10
Valores (HEX)	11	AA	A8	37	30	32	30	30	34	30

Figura 5 – Detalhes do pacote enviado ao Raspberry Pi

É importante notar que o envio para o Raspberry Pi é feito devido ao fato que o Raspberry Pi é o nó *Fog* centralizador da *Mist*.

A partir do envio para o Raspberry Pi, seria possível enviar dados para uma *Cloud*, mantendo um histórico dos dados capturados pelo nó sensor. Para os fins desta monografia, o envio de dados para a *Cloud* não é considerado, dado que o foco é apenas na *Mist*.

Tendo isto em mente, ainda é necessário que o usuário possa alterar alguns parâmetros do nó IoT, tal como desligar ou ligar os avisos físicos do mesmo (buzzer e LED), caso, por exemplo, haja um disparo acidental do alarme de incêndio. Nesse caso, o usuário irá realizar esse controle através do site em desenvolvimento no LRG escolhendo a função a ser realizada como de ativar ou desativar o alarme, ou alterar a frequência de envio dos dados capturados.

Os valores das funções, a nível de dados enviados no pacote, são descritos no Quadro 5.

Quadro 5 – Funções possíveis de envio da Fog para a Mist

Função	Decimal	HEX
Ligar/desligar avisos físicos	a	61
Alterar freqüência de envio de dados	b	62

Dessa forma, será enviado um pacote do Raspberry Pi para o nó *Mist*, para que esse depois envie um pacote indicando o comando para o nó IoT e o mesmo realize a função desejada. A Figura 6 mostra um pacote enviado da *Fog* para o nó *Mist*, indicando que os avisos físicos do nó IoT devem ser desligados, enviando o valor "D" para indicar o desligamento.

	Byte ID	Função	Nó Destino		Valor
Bytes	1	2	3	4	5
Valores (HEX)	11	61	AA	A8	44

Figura 6 – Pacote enviado da *Fog* para o nó *Mist* requisitando o desligamento dos avisos físicos do alarme.

Depois desse envio da *Fog* para o nó *Mist*, o nó *Mist* envia um pacote no modo API para o nó IoT, com o conteúdo de dados. No caso do desligamento dos avisos físicos, será enviado o valor "D" (para desligar). Um exemplo disso se encontra na Figura 7.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
7E	00	0F	10	01	00	13	A2	00	40	D9	AA	A8	FF	FE	00	00	44	8D

Figura 7 – Pacote enviado para o nó sensor para desligar avisos físicos.

Dessa forma, o monitoramento dos dados capturados pelo nó IoT pode ser realizado de forma simples e rápida pelo usuário, graças a arquitetura de *Fog Computing* e sua integração com a *Cloud.*

6.1 Estimativa de gasto de energia em um ciclo

Para realizar a captura de dados do nós sensor sobre os valores de fumaça e gás, temperatura e umidade, assim como a existência de chamas, foi criado um programa usando a linguagem C para ser realizado o *upload* do mesmo no microcontrolador da placa Arduino UNO.

Cada um dos sensores possui diferentes tempos de resposta - 2 segundos para o DHT11 (Aosong, 2018), 1 segundo para o MQ-2 e 1 segundo para o sensor de chama. Assim, cada sensor precisa ficar ligado ao menos durante o tempo de resposta, mas não há necessidade de gasto de energia quando os mesmos não estão enviando dados à *Fog*.

Para que a captura de um sensor não dependa do outro, e assim seja gasto menos energia, foi utilizado o conceito de *threads* para realizar a captura de dados através de um

paralelismo pipeline. Com isso foi definido um ciclo mínimo de 2 segundos, devido a limitação do tempo de resposta do sensor DHT11.

Para realizar o cálculo de gasto de energia no protótipo, é preciso utilizar os valores de cada um dos módulos utilizados no mesmo. Os valores de energia dos módulos, por hora e por segundo, se encontram no Quadro 6.

Quadro 6 – Gasto de energia por modulo

Módulo	Gasto por hora	Gasto por segundo
Umidade e Temperatura (DHT11)	0,3mA	0,00008333mA
Gás e Fumaça (MQ-2)	160mA	0,04444mA
Chamas	0,4mA	0,0001111mA
ATMEGA 328P do Arduino (sleep-mode)	0,1µA	0,000000027778mA
ATMEGA 328P do Arduino	3,2mA	0,00088889mA
Antena XBee (transmissão de dados)	33mA	0,0091666mA
Antena XBee (recebimento de dados)	28mA	0,00777778mA
Antena XBee (sleep-mode)	1µA	0,0000002778mA

Sabendo esses valores é possível calcular o ciclo do nó sensor como sendo:

$$Ciclo\ do\ nó\ sensor = captura\ dos\ sensores + gasto\ da\ antena + sleep - mode$$

Sendo que a captura de dados depende do tempo de resposta de cada um dos sensores mais a corrente, além do gasto de energia do Arduino propriamente dito. Considera-se que o Arduino ficará ligado apenas 2 segundos em um ciclo. Ou seja, o valor gasto na captura dos dados do nó sensor é:

$$Captura\ dos\ sensores = (2s * 0{,}0000833mA) + (1s * 0{,}04444mA) + (1s * 0{,}000111mA) + (2 * 0{,}00088889)$$

$$Captura\ dos\ sensores = 0{,}0465mA/ciclo$$

Após a captura dos dados pelo nó sensor, esses são enviados em um pacote até o nó coordenador. Para este calculo, foi considerado um pacote de tamanho médio de 25 bytes (ou 200 bits) e a velocidade de transmissão de 250kbps, a antena gastaria 200/250000 segundos (ou seja, 0,0008 segundos) para o envio desse pacote. Com esse resultado devemos multiplicar o valor de energia gasto pela antena XBee para a transmissão de dados. Com esse calculo, temos o valor gasto de bytes por tempo.

$$Gasto\ da\ antena\ no\ modo\ de\ envio = 0{,}0008 * 0{,}0091666mA$$

$$Gasto\ da\ antena\ no\ modo\ de\ envio = 0{,}00000733328mA/pacote$$

Durante o tempo do ciclo de captura dos sensores, a antena fica ligada em modo recebimento, estando disponível para receber mensagens de controle da *Fog*. A antena, além do envio dos dados, também ficará ligada por um certo tempo durante o ciclo para recebimento dos dados. Assim, o gasto da antena XBee no modo de recebimento – explicitado no Quadro 6 – deve ser multiplicado pelo tempo do ciclo. Com isso, tem-se o valor de energia gasto pela antena XBee durante o recebimento dos dados.

$$Gasto\ da\ antena\ no\ modo\ de\ recebimento = 0{,}007777778mA/s * 2s$$

$$Gasto\ da\ antena\ no\ modo\ de\ recebimento = 0{,}015555556mA/ciclo$$

Considerando o tempo que o nó IoT ficará "acordado" em um ciclo, temos o valor de gasto energético em um ciclo como:

$$Gasto\ acordado = 0{,}062062889mA/ciclo$$

A definição do ciclo faz com que, depois do envio do pacote, não esteja sendo considerado uma mensagem de confirmação de recebimento antes do nó entrar em *sleep-mode.* Depois do envio do pacote, o nó IoT entra em *sleep-mode.* Isso é calculado com os valores do microprocessador e da antena XBee, ambos em *sleep- mode*.

$$Sleep - mode = (0{,}00000002778mA + 0{,}0000002778mA) * T$$

O tempo T do *Sleep-mode* é uma variável gerenciável a ser controlada na *Fog*, de modo a alterar o intervalo do ciclo, podendo ser modificado com o uso de mensagens enviadas para o nó IoT. É preciso ainda comentar que no início do ciclo, a antena XBee é ligada, e a mesma fica no modo recebimento durante o tempo do ciclo. No fim do ciclo é feito o envio dos dados para a *Fog*, e então tanto a antena quanto o Arduino entram em *sleep-mode*. A Figura 8 demonstra essa situação.

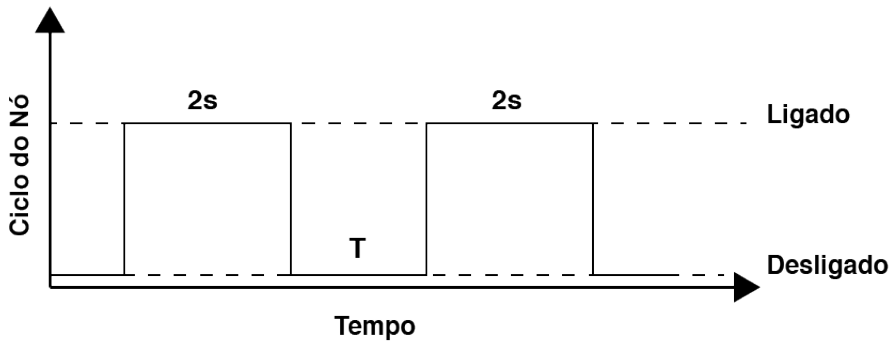


Figura 8 – Gasto de energia do nó IoT no tempo

Existe, porém, um limite associado ao tempo máximo de T. Segundo a instrução normativa IN 012/DAT/CBMSC do Corpo Militar de Bombeiros Estadual de Santa Catarina, foi definido que dispositivos utilizados para alarmes de incêndio possuem um tempo máximo

de resposta para ser aceito dentro dos padrões de segurança estabelecidos em lei. O tempo de resposta máximo para a detecção de chamas por um dispositivo de alarme de incêndio é de 5 segundos (Santa Catarina, 2014). Tendo em mente esse valor, e considerando que o tempo já existente de duração de um ciclo do protótipo IoT é de 2 segundos, o tempo máximo em que o nó pode se encontrar em *sleep-mode* é de 3 segundos. O Quadro 7 faz uma comparação de estimativas de gasto de energia, alterando os valores de tempo T de *sleep-mode* do microprocessador do Arduino e da antena XBee.

Quadro 7 – Gasto energético alterando o tempo T de sleep-mode do nó IoT

	0 segundos	1 segundo	2 segundos	3 segundos
Gasto em sleep-mode	0mA	0,00000030556mA	0,00000061116mA	0,00000091667mA
Gasto acordado	0,062062889mA	0,062062889mA	0,062062889mA	0,062062889mA
Estimativa de gasto total	0,062062889mA/ciclo	0,06206319444mA/ciclo	0,06206350000mA/ciclo	0,06206380556mA/ciclo

Outro valor importante de ser calculado é em relação ao tempo de vida de uma bateria para utilizar o protótipo. Como o foco é a economia de energia, o melhor então seria usar uma bateria como forma de alimentação, ao invés de utilizar o alarme de incêndio ligado sempre na corrente elétrica de uma residência. Baterias de fácil acesso no mercado possuem cerca de 2.000mA, podendo também variar entre 2.100 e 2.500mA cada. O cálculo para estimativa de ciclos que cada bateria consegue durar é a seguinte:

$$Estimativa\ de\ ciclos\ por\ bateria = potência\ bateria / gasto\ ciclo$$

Já a estimativa de tempo em horas, por bateria é calculada da seguinte maneira:

$$Estimativa\ de\ vida\ em\ horas = \frac{(Estimativa\ de\ ciclos\ por\ bateria * tempo\ do\ ciclo)}{3600}$$

O Quadro 8 apresenta a quantidade de ciclos que cada bateria aguenta com o gasto energético do nó IoT, variando também o tempo de *sleep-mode*.

Quadro 8 – Tempo de vida do nó IoT, usando diferentes baterias e considerando os gastos energéticos do protótipo IoT.

	Tempo de vida em ciclos			Tempo de vida em horas		
Tempo T de sleep-mode	**Bateria 2.000mA**	**Bateria 2.100mA**	**Bateria 2.500mA**	**Bateria 2.000mA**	**Bateria 2.100mA**	**Bateria 2.500mA**
0 segundos	32.225,377	33.836,645	40.281,721	17,902	18,798	22,378
1 segundo	32.225,218	33.836,479	40.281,523	26,854	28,197	33,567
2 segundos	32.225,059	33.836,312	40.281,324	35,805	37,595	44,757
3 segundos	32.224,901	33.836,146	40.281,126	44,756	46,994	55,946

É possível notar uma grande diferença de tempo de vida das baterias no período quando se utiliza o tempo de *sleep-mode* do nó IoT, porém ainda assim os resultados ainda não são

considerados ótimos – nos três tipos de baterias utilizadas, nenhuma delas consegue ter tempo de vida acima de três dias. Pode-se afirmar que isso provêm do gasto excessivo do sensor MQ-2, um valor acima mesmo do gasto energético da antena XBee.

6.2 Comparação de gasto de energia usando outros protocolos de rede

Nos cálculos de energia realizados anteriormente, foi considerado que a antena utilizada no nó IoT seria a antena XBee, que por sua vez utiliza o protocolo ZigBee para envio dos dados do nó IoT. Porém, também seria possível utilizar outros protocolos de rede de uso comum, tal como Wi-Fi e Bluetooth; para isso, seria necessário substituir a antena XBee pela antena XBee Wi-Fi ou pelo módulo Bluetooth 4.0 BLE, respectivamente.

O uso de diferentes protocolos de rede para envio de dados do nó IoT torna- se ainda mais difícil se for considerada a possibilidade de que existam mais serviços inteligentes em uma residência, que também utilizem outros protocolos de rede para executar suas funções. Dessa forma, é preciso se atentar ao gerenciamento da *Fog*, pois quanto mais protocolos e serviços diferentes em uma mesma residência, mais complexo ficaria o sistema em relação ao gerenciamento da *Fog*.

A antena XBee Wi-Fi permite envio de dados via Wi-Fi, podendo enviar dados com velocidade de até 72 Mbps (Digi, 2018). Assim como a antena XBee ZigBee, também é possível realizar a configuração da mesma através do *software* XCTU. Os valores de gasto de energia quando utilizando a antena XBee Wi-Fi se encontra no Quadro 9.

Quadro 9 – Gasto de energia da antena XBee Wi-Fi

Módulo	Gasto por hora	Gasto por segundo
Antena XBee Wi-Fi (transmissão de dados)	309mA	0,08583333mA
Antena XBee Wi-Fi (recebimento de dados)	100mA	0,02777777mA
Antena XBee Wi-Fi (sleep-mode)	6µA	0,00000166666mA

Os outros valores envolvidos no cálculo (como gasto do microcontrolador do Arduino e dos sensores) ainda continua o mesmo. Os gastos de energia do nó IoT, caso fosse utilizada a antena XBee Wi-Fi se encontram no Quadro 10, variando os valores de tempo de *sleep-mode* do nó IoT e a potência da bateria utilizada.

Quadro 10 – Tempo de vida de baterias de diferentes potências, utilizando antena XBee Wi-Fi.

	Tempo de vida em ciclos			Tempo de vida em horas		
Tempo T de sleep-mode	Bateria 2.000mA	Bateria 2.100mA	Bateria 2.500mA	Bateria 2.000mA	Bateria 2.100mA	Bateria 2.500mA
0 segundos	19.583,992	20.563,192	24.479,990	10,879	11,423	13,599
1 segundo	19.583,667	20.562,850	24.479,584	16,319	17,135	20,399
2 segundos	19.583,342	20.562,509	24.479,178	21,759	22,847	27,199
3 segundos	19.583,017	20.562,168	24.478,772	27,198	28,558	33,998

Em comparação com o uso da antena XBee ZigBee, o uso do Wi-Fi para envio dos dados não tem valores melhores a oferecer – pelo contrário, o tempo de vida das diferentes baterias é menor do que se comparado com o uso do protocolo ZigBee.

Com o uso do módulo Bluetooth 4.0 BLE é possível realizar o envio dos dados capturados pelo nó IoT com o protocolo Bluetooth, porém sua grande desvantagem é a distancia permitida pelo módulo, de apenas 100 metros dentro de um ambiente fechado (Osoyoo, 2016). Ao contrário das antenas XBee ZigBee e Wi-Fi, o módulo Bluetooth 4.0 BLE não possui diferenciação de gasto de energia quanto a envio e recebimento de dados. O Quadro 11 mostra o gasto energético do módulo.

Quadro 11 – Gasto de energia do modulo Bluetooth 4.0 BLE.

Módulo	Gasto por hora	Gasto por segundo
Bluetooth 4.0 BLE	8.5mA	0,00236111mA
Bluetooth 4.0 BLE (sleep-mode)	1,5mA	0,000416mA

Assim, como na estimativa realizada com o uso da antena XBee Wi-Fi, os outros valores envolvidos no cálculo, como o gasto do microcontrolador do Arduino e dos sensores, ainda continuam os mesmos. Os gastos de energia do nó IoT, caso fosse utilizado o módulo Bluetooth 4.0 BLE se encontra no Quadro 12, variando os valores de tempo de *sleep-mode* do nó IoT e a potência da bateria utilizada.

Quadro 12 - Tempo de vida de baterias de diferentes potências, utilizando o módulo Bluetooth 4.0 BLE.

	Tempo de vida em ciclos			Tempo de vida em horas		
Tempo T de sleep-mode	Bateria 2.000mA	Bateria 2.100mA	Bateria 2.500mA	Bateria 2.000mA	Bateria 2.100mA	Bateria 2.500mA
0 segundos	39.045,553	40.997,830	48.806,941	21,691	22,776	27,114
1 segundo	38.730,479	40.667,003	48.413,099	32,275	33,889	40,344
2 segundos	38.420,449	40.341,472	48.025,562	42,689	44,823	53,361
3 segundos	38.115,344	40.021,111	47.644,180	52,937	55,584	66,172

Quando comparado com os valores obtidos na estimativa de gasto de energia utilizando a antena XBee ZigBee, o uso do módulo Bluetooth 4.0 BLE, e consequentemente o uso do protocolo de rede Bluetooth, apresenta uma maior economia de energia das baterias – ou seja, um maior tempo de vida das baterias, em todos os casos. Isso se deve principalmente ao fato de o módulo não possuir diferentes valores energéticos nos casos de envio e transmissão de dados (como ocorre em ambas antenas XBee ZigBee e Wi-Fi), além de valores menores de gasto no módulo quando acordado.

7. Conclusões e Trabalhos Futuros

Com os conceitos de *Internet of Things* e *Fog Computing* em mente, foi criado um protótipo IoT de alarme de incêndio, onde é possível realizar a captura de dados de fumaça, gás, umidade, temperatura e presença de chamas, e enviar os mesmos para um nó na *Mist*, e consequentemente, para a *Cloud* posteriormente. Assim, é possível realizar o monitoramento em tempo real de uma situação caso seja detectado um cenário de incêndio na residência do usuário.

Quanto a necessidade de economia de energia, foi decidido que o nó IoT usaria uma bateria (ao invés de ser ligado à corrente elétrica da residência) para seu funcionamento. Além disso, foram realizados cálculos do tempo de vida útil dessa bateria, alterando parâmetros de potência da bateria e diferentes protocolos de rede utilizados para realizar o envio dos dados dentro da arquitetura proposta.

Com esses resultados, pode-se notar que o tempo de vida de uma bateria, quando utilizando os sensores DHT11, MQ-2 e sensor de chama, e quando utilizando a antena XBee ZigBee, não conseguiria durar muito tempo (no melhor dos casos, a bateria precisaria ser trocada a cada dois dias). Isso se deve principalmente ao fato de que o sensor de fumaça e gás MQ-2 possui um gasto energético muito alto – maior mesmo do que o gasto da antena XBee ZigBee.

Dessa forma, uma opção para aumento do tempo de vida do nó IoT seria trocar o uso do sensor MQ-2 por outro sensor de gás e fumaça que tenha um valor de gasto energético menor do que o MQ-2.

Outra opção para economia de energia seria utilizar o nó IoT ligado na corrente elétrica para o funcionamento “comum”, e utilizar a bateria apenas para uma situação de falta de energia elétrica. Assim, ainda seria possível utilizar o alarme de incêndio mesmo numa situação de queda de energia.

Quanto a comparação de tempo de vida do nó quando utilizando outros protocolos de rede (Wi-Fi e Bluetooth), é possível perceber que o uso do Wi-Fi não é recomendado para ser o protocolo de envio de dados no caso do protótipo de nó IoT desenvolvido neste trabalho; Devido ao grande gasto de energia da antena XBee Wi- Fi, o tempo de vida do nó diminui ainda mais do que se fosse utilizada a antena XBee ZigBee. Dessa forma, o uso do protocolo de Wi-Fi não seria recomendado.

Quanto ao uso do Bluetooth, os resultados do tempo de vida foram melhores quando comparados com os valores de uso com ZigBee. Porém, vale ressaltar que a distância física possível dentro da residência é menor quando usando ZigBee do que quando usando o Bluetooth. O que deve ser considerado nesse caso também, é que os valores estimados são em relação a dois módulos específicos, e que com outras opções de peças esses valores podem mudar (principalmente em relação ao alcance de cada um dos módulos).

Referências:

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. **The internet of things: A survey**. Computer networks, v. 54, n. 15, p. 2787-2805, 2010. Disponível em: <https://www.researchgate.net/profile/Luigi_Atzori2/publication/ 222571757_The_Internet_of_Things_A_Survey/links/546b36df0cf2f5eb180914e5/Th e-Internet- of-Things-A-Survey.pdf>. Acesso em 17 de Março de 2017.

BIASON, Alessandro et al. **EC-CENTRIC: An energy-and context-centric perspective on IoT systems and protocol design**. IEEE Access, v. 5, p. 6894-6908, 2017. Disponível em: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7895124>. Acesso em 2 de Maio de 2018.

BOTTA, Alessio et al. **Integration of cloud computing and internet of things: a survey**. Future Generation Computer Systems, v. 56, p. 684-700, 2016. Disponível em: <http://wpage.unina.it/valerio.persico/pubs/CloudIoT_FGCS.pdf>. Acesso em 4 de Abril de 2018.

CHEN, Yan-Da; AZHARI, Muhammad Zulfan; LEU, Jenq-Shiou. **Design and implementation of a power consumption management system for smart home over fog-**

cloud computing. In: 2018 3rd International Conference on Intelligent Green Building and Smart Grid (IGBSG). IEEE, 2018. p. 1-5.

DIGI. **Digi XBee Zigbee**. 2018. Disponível em: <https://www.digi.com/products/xbee-rf-solutions/2-4-ghz-modules/xbee-zigbee>. Acesso em 11 de Agosto de 2018.

FRIEDLI, Martin et al. **Energy efficiency of the Internet of Things. Technology and Energy Assessment Report prepared for IEA 4E EDNA.** Lucerne University of Applied Sciences, Switzerland, 2016. Disponível em: <https://www.iea- 4e.org/document/384/energy-efficiency-of-the-internet-of-things-technology-and- energy-assessment-report>. Acesso em 4 de Junho de 2018.

GHAZAL, Bilal et al. **Multi control chandelier operations using XBee for home automation.** In: Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2015 Third International Conference on. IEEE, 2015. p. 107- 111. Disponível em: <https://www.researchgate.net/profile/Bilal_Ghazal/publication/301403907_Multi_control_chandelier_operations_using_XBee_for_home_automation/links/5826e22508ae950ace6c51e5/Multi-control-chandelier-operations-using-XBee-for-home-automation.pdf>. Acesso em 23 de Setembro de 2018.

IORGA, Michaela et al. **Fog Computing Conceptual Model**. 2018. Disponível em: <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-325.pdf>. Acesso em 15 de Abril de 2018.

JALALI, Fatemeh et al. **Greening IoT with fog: A survey.** In: Edge Computing (EDGE), 2017 IEEE International Conference on. IEEE, 2017. p. 25-31. Disponível em: <https://www.researchgate.net/profile/Fatemeh_Jalali2/publication/318418113_Greening_IoT_with_Fog_A_Survey/links/5a1b37080f7e9be37f9bea5f/Greening-IoT-with- Fog-A-Survey.pdf>. Acesso em 7 de Fevereiro de 2018.

MANYIKA, James et al. **Unlocking the Potential of the Internet of Things.** McKinsey Global Institute, 2015. Disponível em: <http://aegex.com/images/uploads/white_papers/Unlocking_the_potential_of_the_Internet_of_Things___McKinsey__Company.pdf>. Acesso em 15 de Abril de 2018.

SAMPAIO, Hugo; MOTOYAMA, Shusaburo. **"Implementation of a greenhouse monitoring system using hierarchical wireless sensor network,"** 2017 IEEE 9th Latin-American Conference on Communications (LATINCOM), Guatemala City, 2017, pp. 1-6. DOI: 10.1109/LATINCOM.2017.8240156

SINGH, Manpreet; KAUR, Ramanpreet. **Integration of IoT and Fog: Need of the Hour.** International Journal of Advanced Research in Computer Science, v. 7, n. 6, 2016. Disponível em: <http://www.ijarcs.info/index.php/Ijarcs/article/view/2741/2729>. Acesso em 4 de Abril de 2018.

STOJKOSKA, Biljana L. Risteska; TRIVODALIEV, Kire V. **A review of Internet of Things for smart home: Challenges and solutions.** Journal of Cleaner Production, v. 140, p. 1454-1464, 2017. Disponível em: <https://www.researchgate.net/profile/Biljana_Risteska_Stojkoska/publication/308975029_A_review_of_Internet_of_Things_for_smart_home_Challenges_and_solutions/links/5af533a34585157136ca43a3/A-review-of-Internet-of-Things-for-smart-home-Challenges-and-solutions.pdf>. Acesso em 23 de Setembro de 2018.

VARGHESE, Blesson et al. **Feasibility of Fog Computing.** arXiv preprint arXiv: 1701.05451, 2017. Disponível em: <https://arxiv.org/pdf/1701.05451.pdf>. Acesso em 4 de Abril de 2018.

ZIMMERMANN, André et al. **Arquitetura para ganho de eficiência energética em redes de sensores sem fios de próxima geração.** 2008. Disponível em: <https://repositorio.ufsc.br/handle/123456789/91383>. Acesso em 30 de Maio de 2018.