UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

TECNÓLOGO EM REDES DE COMPUTADORES

WANDERSON LUAN ARAUJO SAMPAIO

AVALIANDO A EFICIÊNCIA ENERGÉTICA DE UMA CONEXÃO COM A INTERNET ATRAVÉS DO GPRS EM UM CENÁRIO IOT: UM ESTUDO DE CASO COM O SIM800L E O *MIDDLEWARE* DOJOT

QUIXADÁ

2018

WANDERSON LUAN ARAUJO SAMPAIO

AVALIANDO A EFICIÊNCIA ENERGÉTICA DE UMA CONEXÃO COM A INTERNET ATRAVÉS DO GPRS EM UM CENÁRIO IOT: UM ESTUDO DE CASO COM O SIM800L E O *MIDDLEWARE* DOJOT

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Tecnologia em Redes de Computadores do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de tecnólogo em Tecnologia em Redes de Computadores.
Área de concentração: Computação.

Orientador: Prof. Me. Antonio Rafael Braga

QUIXADÁ

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S186a Sampaio, Wanderson Luan.
Avaliando a eficiência energética de uma conexão com a Internet através do GPRS em um Cenário IoT : Um estudo de caso com o SIM800L e o Middleware Dojot / Wanderson Luan Sampaio. – 2018.
60 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Redes de Computadores, Quixadá, 2018.
Orientação: Prof. Prof. Me. Antonio Rafael Braga.

1. Internet das Coisas. 2. Energia-Consumo. 3. General Packet Radio Service. I. Título.

CDD 004.6

WANDERSON LUAN ARAUJO SAMPAIO

AVALIANDO A EFICIÊNCIA ENERGÉTICA DE UMA CONEXÃO COM A INTERNET ATRAVÉS DO GPRS EM UM CENÁRIO IOT: UM ESTUDO DE CASO COM O SIM800L E O *MIDDLEWARE* DOJOT

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Tecnologia em Redes de Computadores do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de tecnólogo em Tecnologia em Redes de Computadores.
Área de concentração: Computação.

Aprovada em: __/__/___

BANCA EXAMINADORA

Prof. Me. Antonio Rafael Braga (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Marcio Espíndola Freire Maia
Universidade Federal do Ceará (UFC)

Prof. Dr. Paulo Armando Cavalcante Aguilar
Universidade Federal do Ceará (UFC)

A Deus

AGRADECIMENTOS

Agradeço à minha mãe, Maria Leoneide, e ao meu pai Carleone Soares, por todo apoio e educação que foi dada permitindo meu ingresso na Universidade, ao Prof. Antonio Rafael Braga por toda orientação e paciência que teve comigo durante a conclusão desse trabalho.

A todo o pessoal da R.H por estarem comigo e presentes na minha vida acadêmica e pessoal durante todo esse tempo.

Ao meu grande amigo Alisson Lima por te me ajudado inúmeras vezes e pelo apoio muitas vezes emocional que em alguns momentos eu necessitei.

A minha amiga Juliana Castro por sempre me ajudar e fazer com que eu mantivesse o foco e os pés no chão, sem seus incentivos e puxões de orelha essa jornada não seria possível. A todos os meus amigos que estiveram comigo desde 2013, que são pessoas sensacionais, os levarei no meu coração para sempre.

A Manuella, uma pessoa muito especial, que me ajudou e me deus conselhos nos últimos 4 anos, que me ajudaram a fizeram este dia ser possível. Ao meu amigo Ari, por ter me emprestado alguns equipamentos necessários para a conclusão deste trabalho, e a sua filha Joyce pelos conselhos.

A minha querida amiga Roseli e meu amigo Cleber, por disponibilizar de tempo para me ajudar nas correções desta monografia.

As professores que estiveram na minha banca, Prof. Dr. Márcio Espíndola Freire Maia e Prof. Dr. Paulo Armando Cavalcante Aguilar pela disponibilidade para participar da banca e pela avaliação realizada.

A todos que estiveram presentes diretamente e indiretamente na minha formação.

“Sentimentos são como rochas que são construídas no nosso coração. Seriam essas rochas os restos de ontem ou os fundamentos do amanhã?”

(Lelouch Lamperouge)

RESUMO

Internet das Coisas compõe um novo paradigma, que tem como objetivo conectar diferentes objetos usados no dia a dia à rede mundial de computadores, sendo eles físicos ou virtuais de modo que interajam e troquem informações entre si de forma autônoma. Uma das principais necessidades que surge com o avanço da Tecnologias da Informação é a busca por informações, diante disso, sistemas de monitoramento são fundamentais para obtenção de informações e entendimento de eventos gerados por determinado ambiente, afim de que ações possam ser tomadas a partir dessas informações. Um fator crucial dentro do cenário de IoT é a comunicação entre esses dispositivos. Uma opção seriam as redes sem fio pois apresentam inúmeras vantagens, dentre elas, sua implementação rápida e de baixo custo. Entretanto, a comunicação por redes sem fio é um meio bastante sensível a interferências (por exemplo: paredes e árvores), dificultando a a troca de informações entre esses dispositivos. Outro fator que deve ser levado em consideração é a autonomia do sistema de monitoramento. Diante disso, esse trabalho apresenta um sistema de monitoramento remoto baseado em Internet das Coisas que pode ser aplicado em diferentes cenários onde é utilizado a tecnologia GSM/GPRS para o envio das informações coletas no ambiente em tempo real. O sistema tem como intuito integrar uma plataforma de IoT com a utilização da tecnologia GSM/GPRS, além de identificar se essa tecnologia pode ser viável para ambientes de monitoramento.

Palavras-chave: IoT. Monitoramento. GSM/GPRS. Consumo Energético.

ABSTRACT

Internet of Things composes a new paradigm, that has objective is to connect differents objects used daily to the world wide computer network, being they physical or virtual so that they interact and exchange information among themselves in an autonomous way. One of the main needs that emerges with the advancement of Information Technology is the search for information, therefore monitoring systems are fundamental to obtain information and understanding of events generated by a given environment, so that actions can be taken from these information. A crucial factor within the IoT scenario is the communication between these devices, an option that arises are the wireless networks that have as main advantages to be implemented quickly with low cost, but it presents as a disadvantage to be a means of communication sensitive to interference (for example: walls and trees) that may hinder communication between these devices, another factor that must be taken into account is the autonomy of the monitoring system. From this, this article presents a remote monitoring system based on Internet of Things that can be applied in different scenarios where GSM/GPRS technology is used to send information collected in the environment in real time. The system is intended to integrate an IoT platform with the use of GSM/GPRS technology, as well as to identify if this technology can be viable for monitoring environments.

Keywords: IoT. Monitoring. GSM/GPRS. Power Consumption.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Ah	*Ampère-hora*
API	*Application Programing Interface*
CPqD	Centro de Pesquisa e Desenvolvimento em Telecomunicações
ETSI	*European Telecommunications Standards Institute*
FUNTTEL	Fundo para o Desenvolvimento Tecnológico das Telecomunicações
GPRS	*General Packet Radio Service*
GSM	*Global System for Mobile Communications*
IoT	*Internet of Things*
MQTT	*Message Queuing Telemetry Transport*
RFID	*Radio-Frequency IDentification*
SMS	*Short Message Service*
WLANs	*Wireless Local Area Network*

SUMÁRIO

1 INTRODUÇÃO

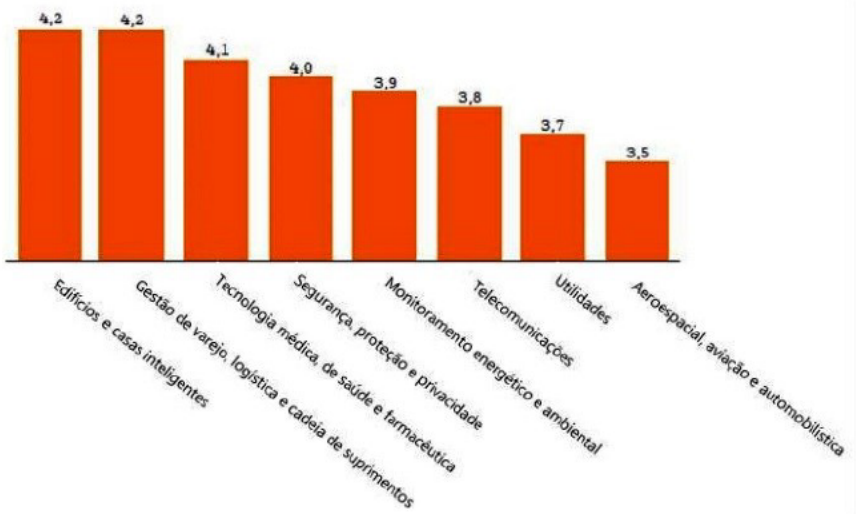
O termo Internet das Coisas (IoT) surgiu em 1999 quando Kevin Ashton realizou uma apresentação na empresa Procter & Gamble (P&G). (ASHTON, 2009) falando sobre a ideia de se etiquetar eletronicamente os produtos da empresa para facilitar a logística da cadeia de produção, através de identificadores de rádio frequência *Radio-Frequency IDentification* (RFID). Dez anos mais tarde escreveu o artigo "A Coisa da Internet das Coisas" para o RFID *Journal*. Segundo Ashton (2009) a "Internet das Coisas" se refere a uma revolução tecnológica.

Internet das Coisas compõe um novo paradigma, seu objetivo é conectar diferentes objetos usados no dia a dia à rede mundial de computadores, sendo eles físicos ou virtuais de modo que interajam e troquem informações entre si de forma autônoma, a fim de atingir um objetivo comum, independente de *hardware* e *softwares* utilizados.

De acordo com Darnell (2015), A Internet das Coisas seria basicamente a prática de conectar, monitorar e gerenciar suas coisas "dispositivos" de forma remota e autônoma, provocando um grande avanço na capacidade de coletar, analisar e distribuir dados, que podem ser transformados em informações e ações.

Dentre as áreas de aplicações baseadas em *Internet of Things* (IoT), destacam-se as que estão voltadas para questões ambientais, condições climáticas, automação residencial, saúde entre outras. Na Figura 1 pode ser observado alguns cenários onde IoT pode ser empregada, com um índice de concentração de aplicações por área, com escala de 1 ate 5, sendo 1 insignificante e 5 muito importante Lacerda (2016).

Figura 1 – Áreas de concentração para Internet das Coisas



Fonte: Lacerda (2016).

Uma das principais necessidades que surge diante de todas esses áreas de aplicações é um sistema que seja capaz de gerenciar todas as informações geradas afim de auxiliar na compreensão de eventos gerados por determinado ambiente de forma a facilitar a adoção de ações preventivas, adaptativas e corretivas L e Filho (2018). A adoção de um sistema de monitoramento em tempo real se torna de grande valia.

Dentro do cenário de IoT, a comunicação entre os dispositivos pode acontecer por diferentes meios de comunicação. Afim de estabelecer essa comunicação, as Redes Sem Fio surgem como uma boa opção, uma vez que apresenta como vantagens: não depender de uma infraestrutura preexistente, requerer configuração mínima, além de ser implantada rapidamente com baixo custo. Os diferentes meios de transmissão incluem Redes de área local sem fio *Wireless Local Area Network* (WLANs) , Bluetooth (IEEE 802.15.1), ZigBee (IEEE 802.15.4), *General Packet Radio Service General Packet Radio Service* (GPRS), entre outros Oliveira e Giglio (2018).

Em contra partida, uma das principais desvantagens de ser utilizar Redes Sem Fio para transmissão de dados em ambientes de monitoramento, se dá pelo fato de ser uma tecnologia sensível a interferência Kridi (2014), onde a quantidade de obstáculos (árvores, paredes) e fenômenos atmosféricos podem dificultar ou até mesmo impossibilitar a comunicação, como no trabalho de Silva (2016) onde após inúmeras tentativas de envio das grandezas coletadas, utilizando uma rede sem fio para o monitoramento de colmeias, não foi possível a comunicação, devido a grande quantidade de árvores no local, optando-se por utilizar um meio de transmissão cabeado.

Dentro os meios de transmissão já citados acima, a tecnologia móvel, GSM/GPRS é o único sistema com cobertura global, mais utilizada em todo mundo, com cobertura em mais de 100 países (GEORGIEV *et al.*, 2004)

Por ser uma tecnologia de grande uso, a motivação principal seria como essa tecnologia pode contribuir dentro do Cenário de IoT, levando em consideração questões de consumo energético por parte de módulos que utilizam essa tecnologia para transmissão de dados, como o SIM800L usado neste trabalho.

Diante do que foi apresentado, esse trabalho tem como objetivo Elaborar uma analise sobre a viabilidade da utilização da Tecnologia GSM/GPRS utilizando um módulo SIM800L, aplicada em um Cenário de IoT, levando em consideração o consumo energético deste módulo, e busca responder os seguintes questionamentos: essa solução pode trazer vantagens em relação a

qualidade de comunicação a um custo relativamente baixo? Essa tecnologia aplicada dentro de um Cenário IoT através da criação de um sistema consegue manter o mesmo ativo por quanto tempo? O gasto energético do módulo GSM/GPRS utilizado neste trabalho é viável para se utilizar em sistemas de monitoramento?.

O Cenário é um sistema de monitoramento genérico, onde não é levado em consideração a parte de sensoriamento, já que não são utilizados sensores e sim apenas variáveis(valor sensor, temperatura) com seus valores já definidos, que ficam sendo enviados para a internet através do modulo, o principal foca deste trabalho está na comunicação do Módulo GSM/GPRS com a internet e seu consumo.

Em resumo, é desenvolvido um sistema para realizar monitoramento em tempo real utilizando uma plataforma de IoT. Para simplificar a visualização dos dados que serão enviados. Utilizando como tecnologia de envio um módulo GSM/GPRS, afim de identificar se o uso dessa tecnologia pode ser viável dentro do cenário de IoT levando em consideração o consumo deste modulo

Afim de simplificar e trazer maior rapidez na implementação do sistema, utilizamos de uma plataforma voltada pra IoT, onde após um levantamento bibliográfico, a plataforma IoT escolhida foi a Dojot, a mesma é desenvolvida por brasileiros membros do CPqD, tendo como principal vantagem uma interface gráfica fornecendo um textitDashboard para a visualização dos dados, além de ser uma plataforma *open source*.

1.1 Objetivos

Nas subseções a seguir são apresentados o objetivo geral e objetivos específicos deste trabalho.

1.1.1 Objetivo Geral

- Avaliar a eficiência energética da transmissão de dados via GSM/GPRS em um cenário de IoT.

1.1.2 Objetivo Específicos

- Desenvolver um Sistema de monitoramento para ser o cenário de testes utilizando uma plataforma IoT *middleware*.

- Utilizar tecnologia de transmissão de dados sem fio, no caso Tecnologia GSM/GPRS.
- Realizar uma analise da transmissão de dados via GSM/GPRS utilizando um módulo GSM/GPRS dentro de um Cenário de IoT.

1.2 Organização

Este trabalho está organizado da seguinte forma. O Capítulo 2 apresenta a Fundamentação teórica abordando conceitos de Sistemas de monitoramento, Rede GSM/GPRS, Internet das Coisas, Dojot e MQTT. Já Capítulo 3 mostra apresenta os trabalhos relacionados. No Capítulo 4 é apresentado a Arquitetura do Sistema onde é aplicada a analise. O Capítulo 5 descreve todos os procedimentos metodológicos necessários para o desenvolvimento da proposta do trabalho e resultados obtidos, O Capitulo 6 apresenta o serviço de Alerta, Por fim, o capítulo 7 temos as considerações finais

2 FUNDAMENTAÇÃO TEÓRICA

Neste capitulo é apresentado uma visão geral sobre todos os conceitos-chaves fundamentais para um melhor entendimento do que é abordado nesse trabalho. Na Seção 3.1 é apresentado uma visão sobre monitoramento ambiental, 3.2 Sistemas de Comunicação Móveis e Rede GSM, 3.3 Conceitos de Internet das Coisas seguida pela seção 3.4 abordando conceitos sobre a plataforma IoT Utilizada, finalizando como o protocolo MQTT, que é utilizado pelo Sistema para publicar dados na plataforma.

2.1 Sistemas de Monitoramento Ambiental

De acordo com Petts (2009), monitoramento é em essência a coleta de dados com o propósito de obter informações sobre uma característica e/ou comportamento de uma variável ambiental. Para esta finalidade, o monitoramento normalmente consiste de um programa de repetitivas observações, medidas e registro de variáveis ambientais e parâmetros operacionais em um período de tempo para um propósito definido. Com o desenvolvimento da tecnologia da informação, os sistemas de monitoramento modernos fornecem dados em tempo real, e através do auxilio destas informações, medidas preventivas e corretivas podem ser tomadas (SILVA, 2016).

De acordo com (OLIVEIRA; RODRIGUES, 2011) aplicações de monitoramento ambiental podem ser classificadas em duas categorias, sendo elas: monitoramento interno (Casas, Edifícios e Locais Fechados) e externo (Monitoramento de Habitats, Tráfego, Locais abertos). Um dos principais desafios encontrados neste tipo de monitoramento, principalmente em áreas naturais, são a autonomia e a baixa intrusão.

Neste contexto, as Redes de Sensores Sem Fio constituem uma alternativa eficiente para o monitoramento ambiental (ALIPPI *et al.*, 2011; OLIVEIRA; RODRIGUES, 2011), onde dois requisitos fundamentais devem ser atendidos, sendo eles a autonomia dos dispositivos tanto de sensoriamento como de comunicação.

Sistema de monitoramento muitas vezes são empregados em regioes remotas e de dificil acesso, onde manutenções na Rede de sensores se tornam inviáveis ou relativamente caras (NAUMOWICZ *et al.*, 2010).

2.2 Sistema de Comunicação Móveis - Redes GSM

A Rede GSM é vista como um sistema de comunicação móvel de segunda geração (2G), uma vez que o sinal e os canais de voz são digitais. Desenvolvido pela *European Telecommunications Standards Institute European Telecommunications Standards Institute* (ETSI), o GSM é o antecessor das tecnologias GPRS e da tecnologia 3G LEDESMA (2015). O GSM opera em várias bandas de frequências não licenciadas, sendo as principais: 900 MHz e/ou 1800 MHz (usados em Europa, Ásia e África) e 1900 MHz (usado na América)(MOULY *et al.*, 1992).

A tecnologia GSM/GPRS é uma integração das redes *Global System for Mobile Communications* (GSM) (*Global Service for Mobile*) e o serviço GPRS (*General Packet Radio Service*), ou seja, é uma extensão da rede GSM, que possui abrangência global Tateoki *et al.* (2007). Os dispositivos capazes de utilizar os serviços GPRS são denominados Modems, podendo utilizar os serviços GSM e GPRS simultaneamente (Classe A), os dois serviços de forma não simultânea (Classe B), ou somente o serviço GPRS (Classe C) (LEDESMA, 2015).

A rede GPRS foi elaborada para disponibilizar tráfego de dados por pacote na rede GSM, esse serviço trouxe o beneficio á rede de telefonia celular de ser interligada a internet. A arquitetura GPRS utiliza a mesma infraestrutura básica utilizada nas redes GSM, na qual inclui novos elementos de rede e interfaces e modifica alguns componentes já existentes (LEDESMA, 2015).

A cobrança pelo uso do GPRS é feita por quantidade de dados transmitidos, enquanto que no GSM é feita por tempo de conexão, o que reduz custos e o torna atrativa em muitas aplicações voltadas pra monitoramento Tateoki *et al.* (2007), como no caso do sistema proposto neste trabalho.

2.2.1 Módulo GSM/GPRS (SIM800L)

Neste trabalho foi utilizado o módulo GSM/GPRS SIM800L, o mesmo pode ser visto na Figura 2. O módulo funciona com tensão miníma de 3.5V, possui a tecnologia *quad-band* embutida, isso implica que o módulo suporta quatro bandas de frequência diferentes para GSM: 850 MHz, 900 MHz, 1800 MHz, (usados em Europa, Ásia e África) e 1900 MHz (usado na América). Segundo LEDESMA (2015), isso implica que o módulo é compatível com praticamente todas as Redes GSM em qualquer parte do mundo.

Através desse módulo é possível conseguir várias formas de comunicação como, por

exemplo, envio de mensagens de texto (SMS), ligações de voz e de dados, e acesso a redes de dados, através do serviço GPRS.

Figura 2 – Módulo SIM800L

Fonte: Elaborada pelo autor.

2.2.2 Pinagem e ligação

O estudo foi realizado utilizando o envio de dados, com apenas os pinos respectivos ao lado esquerdo como pode ser visto na Figura 3, já que os pinos referentes ao lado direito são para realização de chamadas de voz, a seguir uma breve descrição de cada pino.

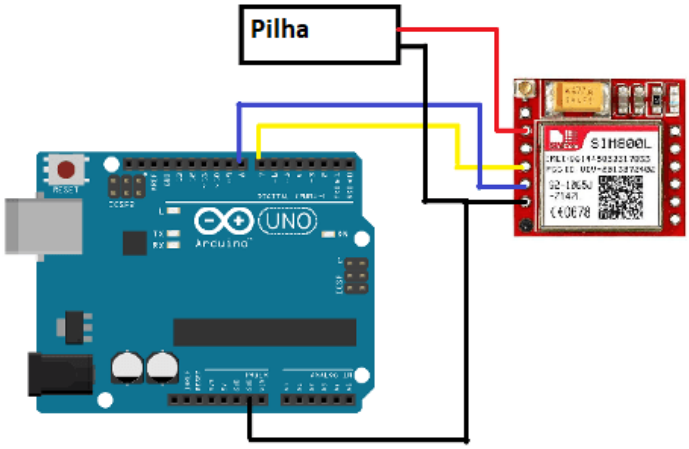
- IPX ANT: Esse *plugins* é dedicado para uma antena externa, o NET também tem o mesmo papel, porém ele é dedicado a antena que vem junto ao módulo, a vantagem de usar o IPX ANT é a melhoria de sinal e alcance;
- Vcc e GND: São pinos para corrente elétrica, de modo a alimentar o módulo, onde o Vcc é o pino positivo que será conectado ao polo positivo da pilha, enquanto o GND é o pino negativo que será conectado ao polo negativo da pilha;
- TXD E RXD: São pinos usados para transmissão e retransmissão das informações.

A seguir podemos observar de forma ilustrativa na Figura 4 a ligação entre o Microcontrolador Arduino + módulo SIM800L.

Figura 3 – Módulo SIM800L (Pinagem)

Fonte: Advice (2018)

Figura 4 – Conexão Arduino + módulo SIM800L



Fonte: Baseada em Mota (2017)

2.3 Internet das Coisas

O termo Internet das Coisas, do inglês *Internet of Things* (IoT) surgiu em 1999 quando Kevin Ashton realizou uma apresentação na empresa Procter & Gamble (P&G). Segundo (ASHTON, 2009),

> "Se tivéssemos computadores que soubessem de tudo o que há para saber sobre coisas, usando dados que foram colhidos, sem qualquer interação humana,

> seríamos capazes de monitorar e mensurar tudo, reduzindo o desperdício, as perdas e o custo. Gostaríamos de saber quando as coisas precisarão de substituição, reparação ou atualização, e se eles estão na vanguarda ou se tornaram obsoletas".

A Internet das Coisas refere-se à integração de objetos físicos e virtuais em redes conectadas a Internet, permitindo que "coisas" coletem, troquem e armazenem informações a respeito do meio que estão inseridas independente de *hardware* e *softwares* utilizados, isso implica em uma enorme quantidade de dados gerados, em que uma vez processados e analisados, geram informações e serviços em escala inimaginável.

Apontada como uma revolução tecnológica iminente e com mercado mundial estimado em 1,7 trilhão de dólares em 2020 (FRAMINGHAM, 2015), a IoT gera impacto em todas as áreas, incluindo eletrônica de consumo, saúde, e de maneira transversal, na forma como a sociedade consome informação (ALMEIDA, 2015).

Essas informações podem ser acessadas a qualquer momento através da Internet usando tecnologias de rede como: *Radio Frequency Identification* (RFID), Wi-Fi, WAN, *Long Term Evolution* (LTE). No entanto, nesses objetos não estão incluídos apenas os dispositivos eletrônicos ou os produtos com maior desenvolvimento tecnológico, como veículos e equipamentos, mas também se incluem coisas como alimentos, roupas, materiais, matérias primas, e outras (RAO *et al.*, 2012). Dessa forma, a IoT compõe-se de dispositivos heterogêneos conectados a rede.

Devido a grande quantidade de dispositivos heterogêneos, derivados de diferentes tecnologias de *hardware* e *software*, faz com que o desenvolvimento de aplicações para IoT envolva uma série de desafios, tais como: heterogeneidade de dispositivos e escalabilidade. Desta forma, plataformas *middleware* tem surgido como solução para prover tal interoperabilidade. Tais plataformas são inseridas entre as aplicações e a infraestrutura (de comunicação, processamento e sensoriamento) subjacente, provendo um meio padronizado para o acesso aos dados e serviços fornecidos pelos objetos através de uma interface de alto nível (BANDYOPADHYAY *et al.*, 2011). A adoção de uma plataforma de *middleware* também pode contribuir para facilitar a construção de aplicações para IoT através de ferramentas Pires *et al.* (2015).

A plataforma de *middleware* foi escolhida levando em consideração requisitos de escalabilidade, suporte local e usabilidade, e ser *open source* . Para o nosso trabalho o *middleware* escolhido foi o DOJOT, sendo melhor descrito na seção 3.4

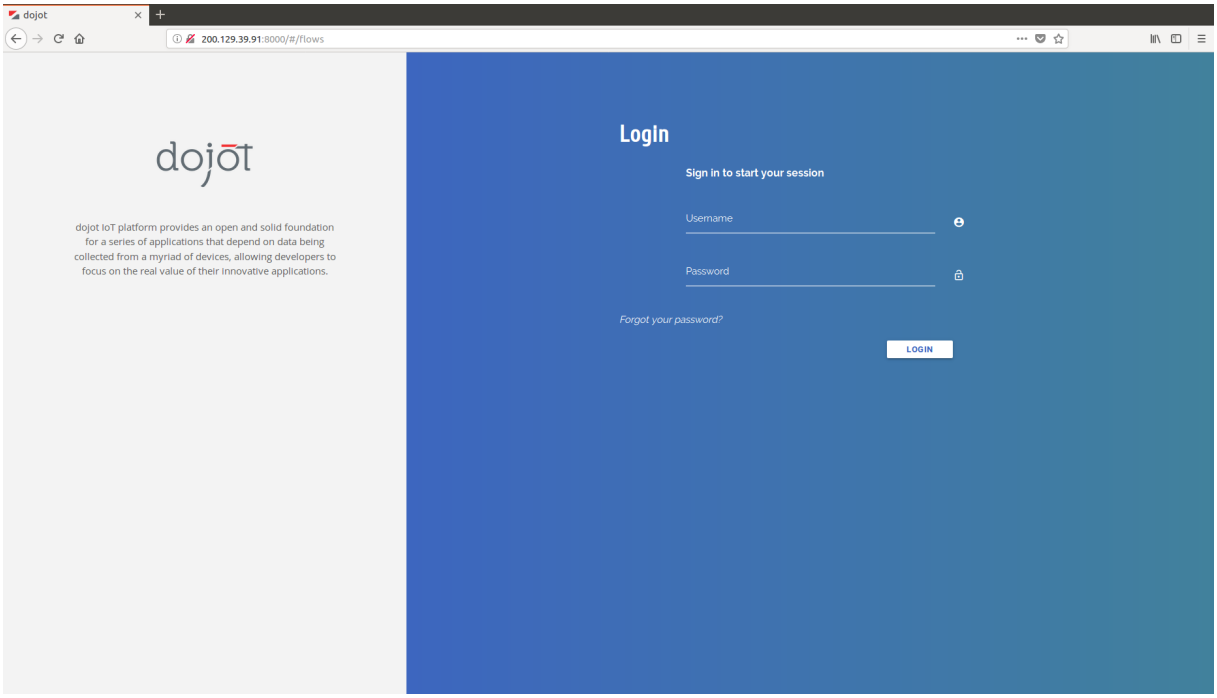
2.4 DOJOT

É uma plataforma de *middleware* aberta lançada pelo CPqD (Centro de Pesquisa e Desenvolvimento em Telecomunicações) onde é levando em consideração robustez e facilidade de uso para o desenvolvimento de aplicações de Internet das Coisas. De acordo com Sahão (2017), o presidente do CPqD afirmou que "a Dojot é uma plataforma habilitadora, capaz de acelerar o desenvolvimento de aplicações IoT adequadas à realidade brasileira, em diversas áreas, e com suporte local."

A plataforma Dojot tem como base o *Fiware*, projeto também *open source* desenvolvido pela União Europeia, sendo um *framework* que consiste em um conjunto de ferramentas e componentes disponíveis para uso, que necessitam de prévios conhecimentos para serem utilizados.

A Dojot possui uma arquitetura baseada em microsserviços e inclui entre outros recursos, um painel de controle com interface gráfica *web* como pode ser visto na Figura 5, uma *Application Programing Interface* (API) (*Application Programing Interface*) única e aberta, para acesso aos serviços, e segurança fim a fim – que envolve a autenticação dos *devices* e das aplicações na plataforma, a gestão de identidade e criptografia – com garantia de privacidade das informações (NICOLAU, 2018).

Figura 5 – Interface gráfica *web*

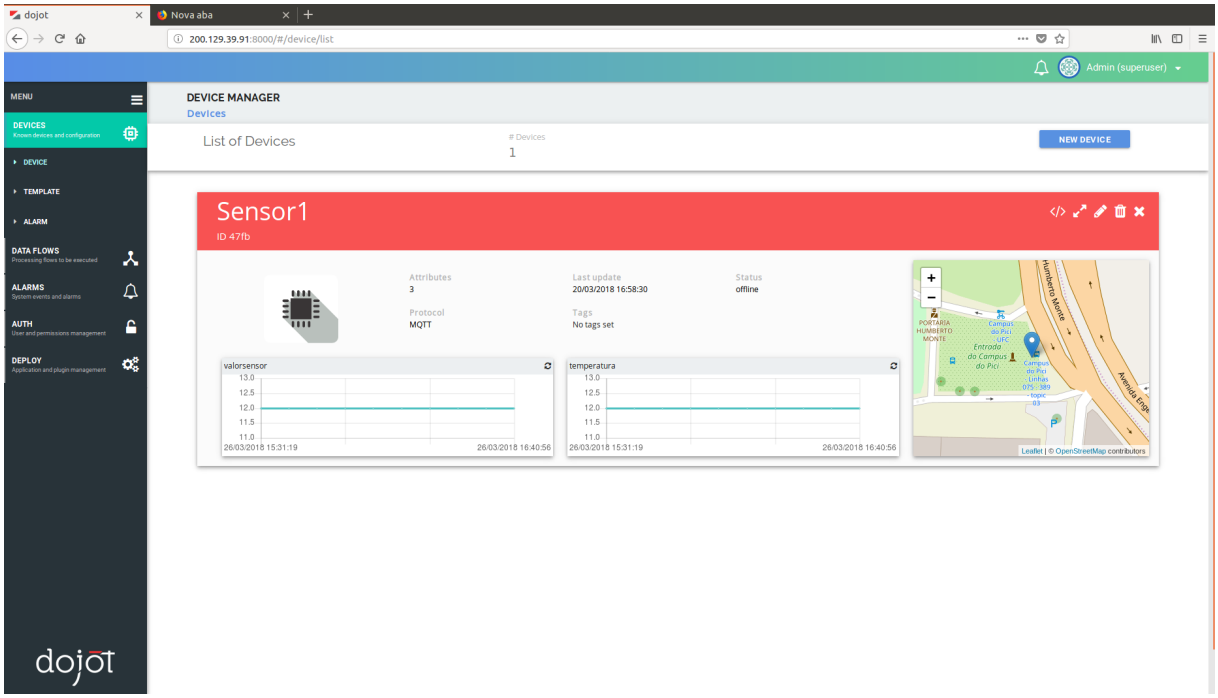


Fonte: Elaborada pelo autor.

A plataforma oferece armazenamento de grandes volumes de dados em diferentes formatos, componentes para análise de serviços em tempo real, suportando os protocolos MQTT e HTTPs. Possibilita construção de fluxos de dados e regras de forma visual, permitindo a rápida prototipação e validação de cenários de aplicações IoT, além de um processamento de eventos em tempo real aplicando regras definidas pelo desenvolvedor Dojot (2018).

Um dos diferenciais da Dojot é o recurso de prototipação rápida, que permite gerar, em cerca de meia hora, um Produto Mínimo Viável (MVP, de *Minimum Viable Product*) a partir do qual é possível validar o conceito junto ao cliente, antes de desenvolver a aplicação (MARIOTE, 2017). Essa função está disponível através de uma interface gráfica onde temos um painel de controle. "Nesse ambiente o usuário também pode criar dispositivos IoT físicos e virtuais, gerenciar fluxos e as aplicações". Mariote (2017) ressalta ainda outro diferencial importante da Dojot: a arquitetura adequada à implantação em ambiente de nuvem. A seguir na Figura 6 é apresentado um sensor criado na plataforma com alguns atributos *online* e sua localização.

Figura 6 – Sensor *online*



Fonte: Elaborada pelo autor.

O desenvolvimento da plataforma Dojot faz parte de um projeto mais amplo, financiado pelo Fundo para o Desenvolvimento Tecnológico das Telecomunicações (FUNTTEL), do Ministério da Ciência, Tecnologia, Inovações e Comunicações (MCTI), e Finep, que vem sendo conduzido pelo Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPqD) em par-

ceria com outras instituições de ciência e tecnologia brasileiras Nicolau (2018).

Como a Dojot possui uma arquitetura baseada em microsserviços, onde é dividida em vários componentes, o componente utilizado neste trabalho foi o GUI. Esse componente implementa uma interface *web* para o usuário gerenciar e configurar funcionalidades básicas do Dojot. Optamos por utilizar esse componente para acelerar todo o processo de gerenciamento e visualização dos dados gerados pelos sensores.

2.5 *Message Queuing Telemetry Transport* (MQTT)

O MQTT é um protocolo de aplicação leve, baseado na topologia *publish/subscribe* para envio de mensagens. Muito aplicado em cenários de comunicação remota entre dispositivos onde a quantidade de dados comunicados é reduzida e a largura de banda para comunicação é limitada (HUNKELER *et al.*, 2008). Diversas linguagens de programação proporcionam suporte às bibliotecas MQTT, incluindo linguagens para dispositivos embarcados com ou sem sistema operacional. Tal versatilidade e compatibilidade torna o protocolo MQTT uma ótima escolha para comunicação de dispositivos IoT.

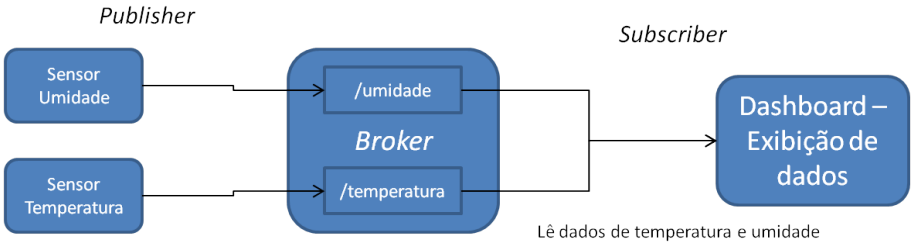
O principio de funcionamento do protocolo parte da utilização de um Servidor MQTT, também conhecido como *Broker* MQTT. Este servidor implementa estruturas de armazenamento de dados, referenciadas como tópicos. Cada tópico refere-se a uma estrutura de publicação *publisher* de dados, mas também possibilita a leitura *subscriber* (FIWARE, 2016).

Existem dois tipos de entidades no protocolo MQTT, onde temos um Servidor MQTT ou *Message Broker* como é chamado, e inúmeros clientes, o Broker tem papel de receber todas as mensagens dos clientes e, em seguida repassa essas mensagens para os clientes de destino (YUAN, 2017).

Para exemplificar melhor a topologia *Publish* e *Subscribe* usando o seguinte cenário: Dois clientes desejam publicar dados no broker MQTT, onde o cliente C1 publica dados de temperatura e C2 publica dados de Umidade. Para isso os clientes C1 e C2 necessitam efetuar publicações em tópicos MQTT, que devem ser previamente definidos. Em geral, cada tópico do mesmo contexto (mesmo usuário) permite a publicação de dados de um único tipo. Desta forma, cada cliente deverá criar um tópico distinto. Na Figura 7, foram criados os tópicos /temperatura e /umidade Fiware (2016).

Com os dados publicados, um terceiro cliente deseja obter os dados referentes a temperatura e umidade, para exibição em uma ferramenta gráfica. Para isso, o terceiro cliente

Figura 7 – Topologia *Publish* e *Subscribe*



Fonte: Fiware (2016).

necessita apenas efetuar a inscrição nos tópicos dos dois outros clientes, e então já estará recebendo as notificações de publicação.

3 TRABALHOS RELACIONADOS

Nesta Seção é apresentado alguns trabalhos relacionados a Sistemas de Monitoramento Ambiental e Residência, onde alguns abordam e usam conceitos de Internet das Coisas.

Ma *et al.* (2011) Desenvolveu um sistema de monitoramento para ajudar agrônomos a compreender melhor os modelos de crescimento das plantas, por meio de rede de sensores e da internet, auxiliando a realizar práticas agrícolas mais eficientes. O ambiente de experimentação foi uma estufa, onde eram coletadas as grandezas de temperatura, umidade relativa do ar, umidade do solo e concentração de dióxido de carbono (CO2), essas informações eram coletas pelos nós sensores e enviadas ao *Gateway* através de um microcontrolador ATMega1281 + AT86RF230. O *Gateway*, microcontrolador baseado no processador Motorola MCF5307 tinha o papel de enviar os dados coletados pelos sensores para o servidor de comunicação, que armazenava os dados em um banco de dados. O sistema de apoio à decisão (DSS) como era chamado, continha vários modelos agrícolas, onde analisava o banco de dados e publicava orientações relevantes, como irrigação, manejo de pragas e advertências climáticas desastrosas, para os agricultores via SMS, todas as informações poderiam ser visualizas em tempo real através de uma página Web.

No trabalho atual não é abordado a parte de sensoriamento, já que é um sistema genérico, o envio das informação e feito através do módulo GSM/GPRS, e possui como semelhança a utilização de uma rede sem fio para transmissão dos dados. Na arquitetura do nosso sistema temos um Agente IoT que responsável por receber esses dados enviados pelo módulo GSM/GPRS e publica-los na plataforma que que funciona de forma semelhante a um *Gateway* da arquitetura de Ma *et al.* (2011). Outras vantagens do trabalho atual são a visualização de todas as informações em tempo real, e todos os dados ficarem salvos em um banco de dados, para possíveis analises futuras, outra semelhança é a utilização do serviço de SMS para notificar os usuários, porém no nosso trabalho as notificações são referentes a alterações nas grandezas monitoradas que possuem valores pré-definidos. No estudo de Ma *et al.* (2011) não foi utilizada nenhuma plataforma de IoT. Já no presente trabalho foi usada a plataforma Dojot.

No trabalho de Sônego *et al.* (2017) ele aplica a utilização de Internet das Coisas para eficiência energética afim de minimizar desperdícios energético, através do controle de intensidade da luminosidade em ambientes fechados, utilizando em sua arquitetura um microcontrolador Intel Galileo e módulos XBee, e uma luminária da marca Philips, para controlar a intensidade de luminosidade utilizou-se de um reator HF-R 214-35 TL5 EII tambem fabricado pela empresa Philips.

O componente central do sistema é o Galinux, que está conectado ao módulo Xbee, seu papel é receber as informações dos sensores, processá-las e enviar comandos á luminária para regular a luminosidade, basicamente o sensor mede a intensidade de luz do ambiente e envia as informações ao módulo Xbee, onde o mesmo repassa as informações ao controlador Galilux. O experimento realizou-se um monitoramento em um dia ensolarado, chuvoso e neblinado, apresentando em dias chuvosos uma redução de 37,93%, além da economia, ressalta-se a importância de manter constantes os níveis de luminosidade no ambiente, em um dia nublado registrou-se uma redução de 58,75% em relação à luminária operando em sua potência total, já em dias ensolarados o resultando em uma economia foi de 65,45 % em comparação à sua utilização na potência máxima. Os experimentos tiveram duração de 12 horas com coletas feitas com intervalo de 10 minutos Sônego *et al.* (2017).

No trabalho atual, apresenta uma arquitetura bem semelhante a de Sônego *et al.* (2017), onde existe um microcontrolador Arduíno conectado ao módulo GSM/GPRS que recebe essas informações e envia para o Agente IoT, que é responsável por publicar na plataforma. A escolha do módulo GSM/GPRS se deu pelo fato possuir um menor custo que um módulo XBee, além de oferecer um bom desempenho em termos de confiabilidade e qualidade de comunicação a um custo relativamente baixo, as informações podem ser vistas em tempo real através da plataforma que pode ser acessada de qualquer lugar, diferente do trabalho do Sônego *et al.* (2017) que não disponibiliza essas informações *online*.

Silva (2016) Propôs um Sistema de monitoramento não invasivo de colmeias utilizando conceitos de Internet das Coisas. O sistema realiza o monitoramento em tempo real de colmeias a um baixo custo, com leituras a cada 60 segundos, podendo ser consultadas através de um aplicativo, levando em consideração o armazenamento dos valores coletados. A plataforma de *middleware* utilizada foi a *Fiware*, ela é voltada pra IoT, a escolha foi feita levando em consideração requisitos de interoperabilidade e escalabilidade. A *Fiware* é uma plataforma que auxilia na coleta e e armazenamento dos dados. Para a coleta dos dados são utilizados de sensores conectados ao microcontrolador Arduíno. As métricas coletadas são: temperatura, umidade relativa e ruido. O Arduíno processa os dados lidos pelos sensores e utilizando de um módulo wi-fi NRF24LO realiza a transmitir dos dados coletados para o *Gateway* que é um microcomputador *Beaglebone Black*(BBB) Silva (2016).

O *Gateway* tem o papel de receber os dados coletados pelos sensores e enviar para o *Middleware* (*Fiware*) no formato JSON através de uma mensagem HTTP. Uma das dificuldades

encontradas foi a comunicação entre o Arduíno e o *Gateway* através da comunicação sem fio via sinal de rádio frequência onde depois de inúmeras tentativas não foi possível a comunicação, devido a grande quantidade de obstáculos, como solução foi utilizado um cabo de rede para conseguir a conexão com a internet Silva (2016).

No presente trabalho utilizamos de um módulo GSM/GPRS para comunicação, visando remover a necessidade de utilizar um microcomputador, de modo que o módulo GSM/GPRS se comunique diretamente com o Agente IoT e o mesmo se comunique com a plataforma instanciada na nuvem. Como já foi citado anteriormente, como plataforma de *Midleware* para fazer o armazenamento e processamento de dados escolheu-se a Dojot, por ser uma plataforma brasileira, possuir uma Interface gráfica e ser *open source*.

Nos trabalhos anteriores não foi realizada nenhuma análise em relação ao consumo dos módulos e como isso pode impactar na autonomia do sistema. Além da análise de consumo de energia do módulo, no presente estudo é feita uma análise em relação a escalabilidade da plataforma usada, afim de identificar se ao aumentar o número de sensores a plataforma consegue atender a todas as requisições que são enviadas. A Tabela 1 apresenta um comparativo entre os trabalhos citados.

Tabela 1 – Comparação entre os trabalhos relacionados

Trabalho	Plataforma de IoT	Visualização	Módulo
Ma *et al.* (2011)	Não	*Software*	ATMega1281 + AT86RF230
Sônego *et al.* (2017)	Não	*Software*	Intel Galileo + módulos XBee
Silva (2016)	Sim	Aplicação Móvel	Arduino + módulo NRF24L0
Trabalho Atual	Sim	Aplicação *web*	Arduino + módulo SIM800L

Fonte: Elaborada pelo autor.

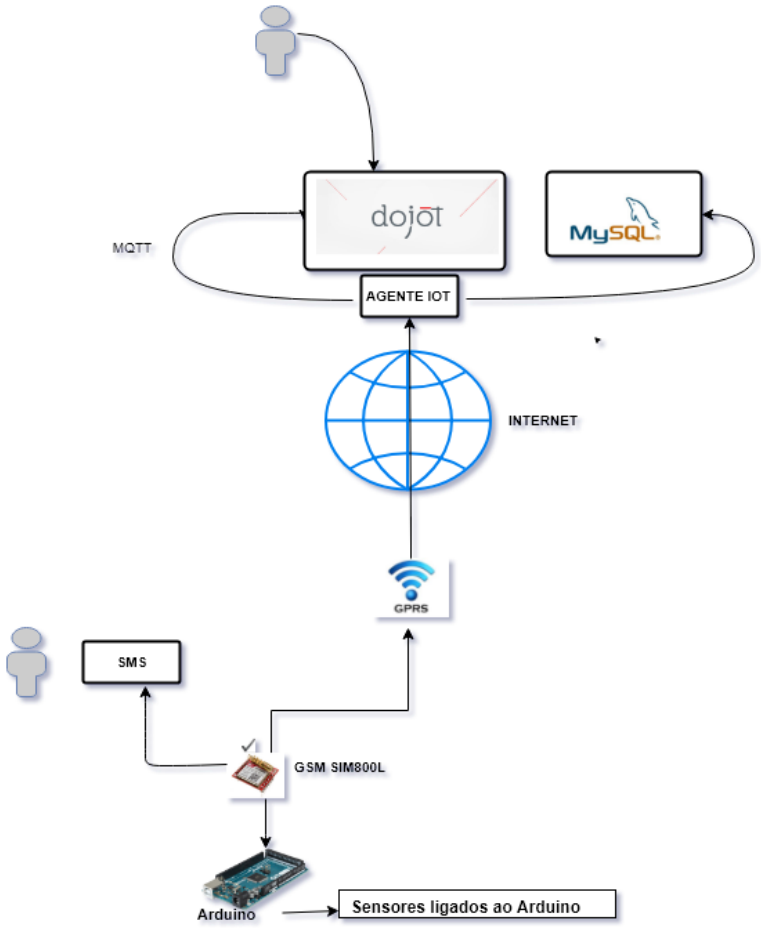
4 APRESENTAÇÃO DO SISTEMA DOJOIOT

Essa seção apresenta o DojoIoT, nome dado ao sistema, onde descreve a arquitetura e seus componentes em detalhes em cada fase desde a coleta até a visualização das informações.

4.1 Arquitetura do DojoIoT

A arquitetura do DojoIoT, apresentada na Figura 8, é composta por 4 elementos principais: Monitoramento, Agente IoT, Banco de dados e *Middleware*. Nas próximas subseções são apresentadas em detalhes todos os elementos da arquitetura.

Figura 8 – Arquitetura do DojoIoT



Fonte: Elaborada pelo autor.

4.1.1 Possibilidade de se utilizar sensores

Os sensores responsáveis pelo monitoramento e coleta de dados poderão ser ligados ao microcontrolador Arduíno, frisando que não foram utilizados sensores neste protótipo, onde os valores enviados foram definidos em variáveis, caso sejam utilizados sensores reais, o Arduíno realizará o processamento dos dados lidos pelos sensores, o mesmo se encontra conectado ao módulo GSM/GPRS SIM800L responsável por transmitir os dados para o Agente IoT utilizando a rede de telefonia celular. As leituras e envios podem ser feitas de forma periódica, sem intervalo e com intervalos de 1 minuto. Nesta arquitetura o módulo tem sua própia fonte de energia.

4.1.2 Agente IoT

O Agente IoT funciona como um webservice, o mesmo é uma aplicação desenvolvida em node js que roda em uma máquina Ubuntu do Laboratório da UFC, o mesmo é responsável por receber os dados enviados pelo Arduino e repassa-los para a plataforma de *Middleware* que é a Dojot. O módulo GSM/GPRS SIM800L é responsável pelo envio ao Agente IoT, a comunicação é feita através de comandos AT, esses comandos são do próprio módulo, permitindo o envio de mensagens HTTP através do método GET. A seguir na Figura 9 mostra um trecho do código de como é feito o envio dos dados coletados através do módulo GSM SIM00L para o Agente IoT.

Figura 9 – Modo de envio do módulo GSM para o Agente IoT

ENVIOHTTPCONCATENADO

```
178  SIM800L.println("AT+HTTPINIT"); //INICIALIZANDO SERVIÇO HTTP
179  delay (2000);
180  resposta();
181
182  SIM800L.println("AT+HTTPPARA=\"CID\",1");
183  delay (2000);
184  resposta();
185
186 //CONCATENANDO OS VALORES PARA REALIZAR O ENVIO
187  String finali="\"";
188  String mensagem="AT+HTTPPARA=\"URL\",\"http://200.129.39.91:3000/?valorsensor=";
189  String concatena= String(mensagem)+ String (valorsensor) + String(meio) + String(temperatura) + String(finali);
190
191 SIM800L.println(concatena);
192 delay (2000);
193  resposta();
194
195  SIM800L.println("AT+HTTPACTION=0");
196  delay (3000);
197  resposta();
198
199  SIM800L.println("AT+HTTPREAD");
200  delay (3000);
201  resposta();
202
203  SIM800L.println("AT+HTTPTERM");
204  delay (1000);
205  resposta();
206
207  SIM800L.println("AT+SAPBR= 0,1"); //EHCERRA
208  delay (2000);
```

Salvo.

Fonte: Elaborada pelo autor.

Após receber os dados que foram enviados pelo módulo GSM SIM800L ao Agente IoT, o mesmo os publica utilizando o protocolo *Message Queuing Telemetry Transport* (MQTT), sendo tratados pelo *broker* da plataforma de *Middleware* Dojot, levando em consideração que o *Middleware* já foi previamente configurado na plataforma com um dispositivo gêmeo. A Figura 10 apresenta um trecho do *script* onde acontece a publicação.

Figura 10 – Agente IoT publicando na plataforma

```
var mqtt = require('mqtt')
var client  = mqtt.connect('mqtt://200.129.39.91')

console.log('inicio');
client.on('connect', function () {
client.subscribe('/admin/47fb/attrs');

setInterval(function()             {
        }, 90000);
//setInterval(function()             {
  client.publish("/admin/47fb/attrs", '{"valorsensor":'+ q.valorsensor +'}');
  //client.publish("/admin/47fb/attrs", '{"umidade":'+ leitura_dht11[1] +'}');
  client.publish("/admin/47fb/attrs", '{"temperatura":'+ q.temperatura +'}');
 //}, 1000);
})
//setInterval(function()             {
//        }, 1000);
client.on('message', function (topic, message) {
  console.log("msg recebida => " + message.toString());
})


}).listen(3000, "200.129.39.91");

sys.puts("Server rodando");
```

Fonte: Elaborada pelo autor.

4.1.3 Dashboard da Dojot

Levando em consideração a facilidade de uso, através do uso da plataforma, podemos facilmente visualizar todos os dados que foram publicados por meio do Agente IoT. A plataforma apresenta varias funcionalidades, como já foi dito na subseção anterior. É necessário que a plataforma tenha sido previamente configurada, isso se refere a criação de dispositivos IoT podendo ser físicos ou virtuais. Então deve ser feito primeiramente a criação desse dispositivo, após a criação do dispositivo, setamos os atributos que vão ser as grandezas monitoradas.

Cada dispositivo criado tem seu ID, esse ID é o identificado e deve ser passado no tópico quando o Agente IoT for enviar os dados para o *Middleware*. A Figura 10 apresenta o ID do dispositivo que corresponde a 47fb, uma das principais vantagens de ser utilizar a Dojot é a rápida criação de dispositivos e atributos.

Todos os dados enviados ao *Middleware* são salvos em um banco de dados MYSQL, de forma a manter um histórico sobre todos os valores coletados para possíveis analises. Por padrão a plataforma utiliza o MongoDB, porém após inúmeras tentativas de configuração para

armazenamento no MongoDB não resultarem em sucesso, optou-se por utilizar o MYSQL.

4.1.4 Serviço de Alerta

Tentando aproveitar o máximo dos recursos disponíveis no módulo GSM/GPRS SIM800L, foi implementado um serviço de notificação via *Short Message Service* (SMS) para alerta caso os valores lidos pelos sensores apresentem valores discrepantes dos padrões pré-definidos. O mesmo é integrado com o *sketch* de envio embarcado no Arduíno, onde uma função fica responsável por notificar o usuário com número telefônico previamente cadastro para receber mensagens SMS. As mensagens podem ser enviadas a cada hora após o envio da primeira, a função pode ser vista no Apêndice B.

4.1.5 Visualização dos Dados

Os dados podem ser facilmente visualizados em tempo real através de um *Dashboard* que a plataforma oferece, apresentando de forma gráfica os valores, o gráfico apresenta valores com intervalos de até 24 horas, desde que o mesmo apresente alguma variação, a plataforma ainda conta com um serviço de autenticação, onde os usuários podem receber níveis de autorização diferentes, sendo eles de usuário comum e admin, cada usuário tem sua própria seção o que implica que um usuário não vai conseguir apagar ou modificar outros dispositivos criados na plataforma.

5 AVALIAÇÃO DO SISTEMA PROPOSTO

A avaliação do sistema foi realizada buscando-se verificar a eficiência do funcionamento as funcionalidades básicas do sistema, tais como envio dos dados e publicação na plataforma, consumo energético do módulo GSM/GPRS SIM800L, além da realização de um teste de escalabilidade medindo o tempo de processamento de acordo com um aumento do número de sensores publicando na plataforma.

Essa seção é dividida em Materiais e Métodos, Experimentos e Resultados, onde a Seção Materiais e Métodos detalha todo material utilizado e implementação realizada para criação do sistema. A seção de Experimento detalha todos os passos para a implementação do sistema no ambiente de testes, e por último, na seção de resultados são mostrados os cenários e os resultados obtidos.

A seguir são descritos todos os materiais e configurações utilizadas para a criação do protótipo de acordo com a arquitetura apresentada na seção 4.1

5.1 Materiais e Métodos

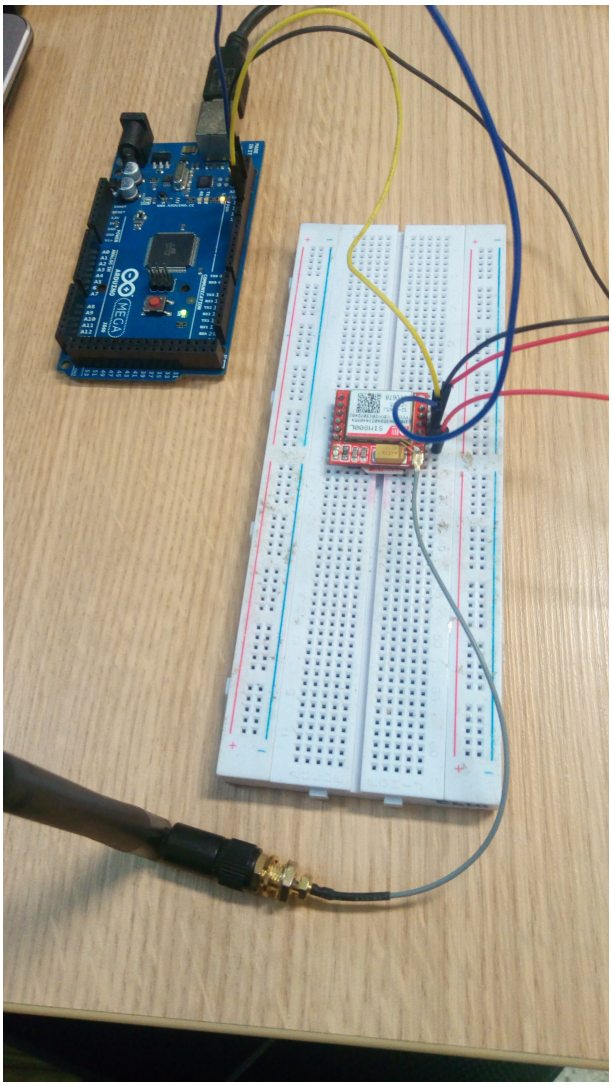
A parte de hardware é composta por um microcontrolador Arduino MEGA 2560 que tem como especificações, uma memória Flash com capacidade de 256 Kbytes, velocidade de clock de 16 MHz e dimensões 101,6mm x 53,4mm. A alimentação do Arduino ocorre pela conexão USB.

O módulo de transmissão utilizado conectado ao Arduino é o SIM800L (Figura 11) + Antena externa de modo a ampliar o alcance do sinal de comunicação, tendo como frequência de operação de 800 à 2600 MHz, um chip celular da operadora claro que fica encaixado na parte inferior do módulo. O módulo é feito para operar com uma voltagem entre 3.4 a 4.4 V, a alimentação e feita através de uma baterial de 12 Volts, com capacidade de 5Ah. Todos os códigos utilizados e embarcados para o Arduino foram codificados em C++ por ser a linguagem nativa da plataforma Arduino. Foi utilizado um multímetro para realizar as medições da pilha.

5.1.1 Formas de Envio SIM800L

Após uma leitura no manual identificamos que havia mais de uma forma de envio, a partir dai surgiu a pergunta: *"Como as diferentes formas de envio podem impactar no consumo energético do módulo SIM800L?"*

Figura 11 – Arduino ligado ao módulo SIM800L



Fonte: Elaborada pelo autor.

Levando isso em consideração, decidiu-se realizar uma análise dessas formas de envio e se isso geraria impacto na autonomia do sistema. A primeira forma de envio é chamada de *TCP/IP Application*, nesse modo, o módulo inicia uma conexão com o webservice, que é o Agente IoT, e transmite essas informações através de uma requisição, sendo elas *GET* ou *POST*, os valores definidos são enviados anexados a URL, a URL é composta por, endereço ip e porta do Agent IoT, seguido pelos valores anexados, após o envio é encerada a conexão.

No modo HTTP *Application*, neste modo, são iniciados os serviços http ou https, neste modo ele aceita apenas requisições GET, não é feito uma conexão previa, onde os valores tambem são anexados a URL e enviados, após o envio, a conexão é encerrada. Ambos os códigos podem ser vistos com mais detalhes no Apêndice A

5.1.2 Comunicação e Visualização dos Dados

O Agente IoT funciona como um webservice, o mesmo foi desenvolvido em JavaScript, ele fica responsável por receber os dados enviados pelo módulo SIM800L, e tem o papel de publica-los na plataforma Dojot. Atualmente devido à problemas na configuração do banco junto com a plataforma, o Agente IoT fica também responsável por salvar todos esses dados em um Banco de Dados MySQL.

Para a instalação da plataforma Dojot, escolhemos uma solução de computação em nuvem, no caso a *Google Cloud Platform*. A decisão se deu pelo fato desse serviço oferecer todos os recursos computacionais necessários por um período de uso gratuito suficiente para a execução do trabalho, todo o processo de instalação e configurações realizados na máquina virtual instanciada na *Google Cloud* podem ser encontrados no Git Hub [1].

5.1.3 Definição dos Cenários

Tentando responder a pergunta feita na seção 5.0.2 foram definidos dois cenários para analisar o consumo do módulo SIM800L, onde foi definido os seguintes fatores: envio usando o modo TCP/IP *Aplication* e envio usando o modo HTTP *Aplication*. Os níveis definidos foram: envio sem intervalo, e com intervalos de 1 minuto, como pode ser visto nas Tabelas 2 e 3, para a realização da analise foi definido o Intervalo de Confiança de 95%.

Tabela 2 – Fatores e níveis definidos

Características	Descrição
Dispositivo	SIM800L
Parâmetros	HTTP *Aplication* e TCP/IP *Aplication*
Níveis	Sem intervalo, 1 minuto
Métricas	Consumo Energético
Técnica de Avaliação	Experimentação
Apresentação dos Resultados	Gráfico

Fonte: Elaborada pelo autor.

5.1.4 Execução

Os experimentos foram realizados na Universidade Federal do Ceará-Campus Quixadá durante os dias 28/05/2018 até o dia 12/06/2018. A Universidade é situada a 5.5 Km do Centro da Cidade, onde a mesma se apresentou como um ótimo lugar para realizar esse experi-

[1] https://github.com/wandersonsampaio/Dojot

Tabela 3 – Cenários de Execução

Cenário	Modo de Envio	Intervalo
1	HTTP *Aplication*	Sem intervalo
2	TCP/IP *Aplication*	Sem intervalo
3	HTTP *Aplication*	1 minuto
4	TCP/IP *Aplication*	1 minuto

Fonte: Elaborada pelo autor.

Tabela 4 – Cenários de Execução para tempo de processamento

Cenário	Métrica
1 sensor	Tempo de processamento
10 sensores	Tempo de processamento
30 sensores	Tempo de processamento

Fonte: Elaborada pelo autor.

mento por se encontrar afastada do Centro Urbano, o que aproxima ainda mais a ideia de um sistema de monitoramento remoto. A Universidade é cercada por grandes pedras e possui uma flora relativamente densa, que de certa forma dificulta a utilização da tecnologia GSM/GPRS.

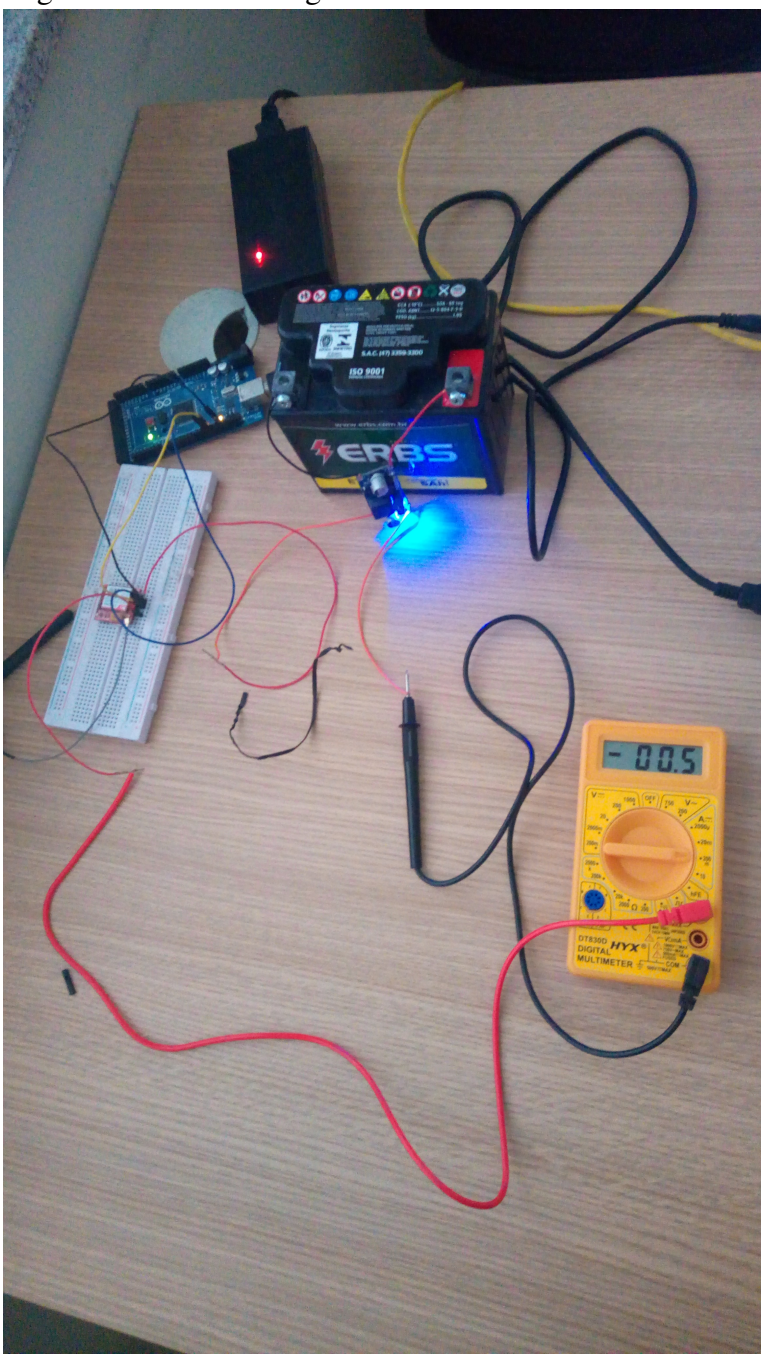
Após embarcar os códigos para o Arduíno, o módulo SIM800L foi ligado a bateria, Os testes realizados foram para medir a corrente gasta pelo módulo durante o tempo de funcionamento e quando estava ocioso afim de conseguir estimar por quanto tempo a bateria conseguiria manter o módulo. Foram realizados dois experimentos de 30 min, um sem intervalos e outro com intervalos de 1 minuto a cada mensagem enviada.

Para validar a analise, os experimentos foram executados 2 vezes. Onde a corrente era medida durante o envio da mensagem, para obter a corrente, utilizamos de um multímetro ligado em serie, de modo que toda a corrente que chega no módulo passasse pelo multímetro.

Para os testes de tempo de processamento, utilizamos de sensores virtuais feitos em JavaScript, cada sensor virtual tinha dois atributos: valor sensor e temperatura. A fim de tentar se aproximar o mais próximo de sensores reais, não foi atribuído valor fixo aos atributos, então foi criado uma função que gerava números aleatórios entre 1 a 100. Os sensores virtuais foram iniciados em uma máquina do Laboratório de Redes da Universidade Federal do Ceará, com as seguintes especificações, Desktop Lenovo Thinkcentre M91p Core I5 2400 3.10ghz, com memória de 7,6 Gb e com o sistema operacional Ubuntu 16.04.

Esses sensores virtuais ficavam publicando na plataforma com intervalos de 5 segundos, onde o fator levado em consideração o tempo de processamento para os sensores publicarem na plataforma. Os níveis definidos foram, 1, 15, e 30 sensores a fim de tentar identificar se de

Figura 12 – Arduino ligado ao SIM800L



Fonte: Elaborada pelo autor.

acordo com um aumento no número de sensores publicando, quanto tempo demora para o sensor publicar na plataforma desde o momento que foi iniciado, no caso, o tempo de resposta.

5.2 Experimentos

Os experimentos a seguir tem como fator duas formas de envio, com níveis sem intervalo de envio e com intervalos de 1 minuto. Cada experimento foi repetido duas vezes com

duração de 30 min por experimento, os valores foram coletados 4 vezes de forma aleatória com o multímetro.

5.2.1 Cenário 1 e 2 sem intervalo de envio

O primeiro Cenário executado foi usando o modo de envio HTTP *Aplication* sem intervalo, onde o tempo para preparar a mensagens e enviar é de 25 segundos. Onde a corrente mostrou uma variação entre 0,01 a 0,09 Ah para o envio de uma mensagem. A corrente média para envio de uma mensagem nesse modo ficou em 0,044 Ah, Durante os 30 min, foram enviadas 140 mensagens, e perdidas 8, que representa uma perda de 5,71%. Com corrente de 0,044 utilizando uma bateria que fornece ate 5 Ah, a estimativa é que o sistema se mantém funcionando por 113,6 horas, que dá em torno de 4,73 dias.

Já usando o modo de envio TCP/IP *Aplication*, sem intervalo, onde o tempo para preparar a mensagens e enviar é de 30 segundos devido este formato ter mais comandos. A corrente mostrou uma variação entre 0,01 a 0,09 Ah para o envio de mensagens. A corrente média para envio de uma mensagem nesse modo ficou em 0,045 Ah, foram enviadas 120 mensagens, e perdidas 5, que representa uma perda de 4,1%.

Ambos os modos de envio praticamente apresentaram a mesma corrente para o envio de mensagens, que fica em torno de 0,04 *Ampère-hora* (Ah), porém o modo HTTP *Aplication* consegue enviar mais mensagens em uma mesma janela de tempo que o TCP *Aplication*, mas acaba perdendo algumas mensagens. Com corrente de 0,045 utilizando uma bateria que fornece 5Ah que é a utilizada neste experimento, o sistema se mantém funcionando por 111,1 horas, que dá em torno de 4,62 dias.

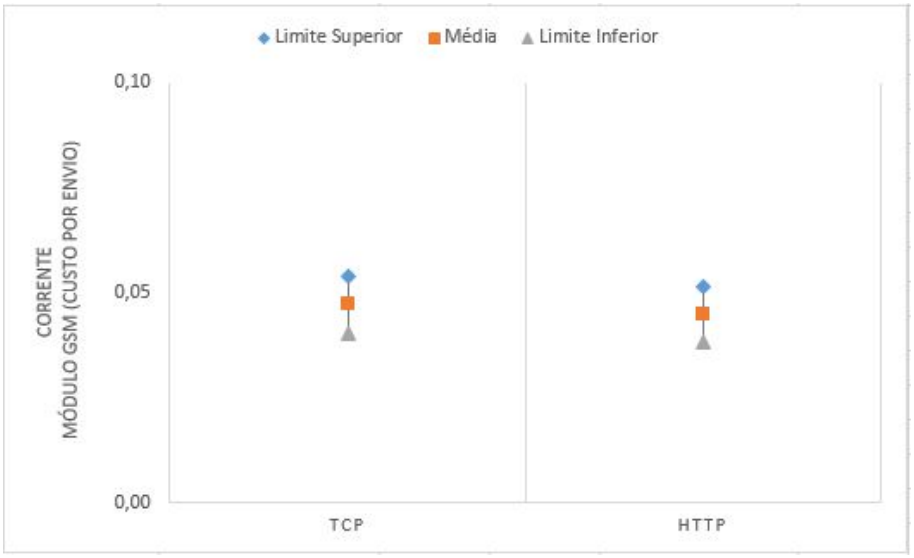
A seguir podemos observar no Figura 13 a média da corrente dos dois modos de envio.

Todas as perdas foram acompanhadas pelo monitor serial, caso tivesse sucesso no envio, o status era 200 indicando que processou com êxito a solicitação feita ao Agente IoT, e 601 caso ocorresse algum erro.

5.2.2 Cenário 3 e 4 com intervalos de envio de 1 minuto

Neste cenário com intervalos de envio de 1 minuto usando o modo HTTP, independente do tempo de intervalo, a corrente vai se manter a mesma para realizar o envio das mensagens, então o que foi observado foi qual a corrente quando o módulo não está enviando,

Figura 13 – Média da corrente por envio de mensagem sem intervalo



Fonte: Elaborada pelo autor.

o módulo consome uma corrente miníma sem está enviando de 0,01 Ah, a partir dai deve ser levado em consideração o tempo de espera e a quantidade de mensagens que vão ser enviadas nesse intervalo de tempo. Levando em consideração que o tempo de envio HTTP *Aplication* é de 25 segundos e TCP/IP *Aplication* é de 30 segundos, e o intervalo é de 1 minuto, foram enviadas no total 40 mensagens para ambos os modos, sendo vinte por experimento, onde tivemos uma perda de 2 mensagens para o HTTP *Aplication* que representa 5,0% de perda, e 1 no modo TCP/IP *Aplication* que representa 2,5% de perda.

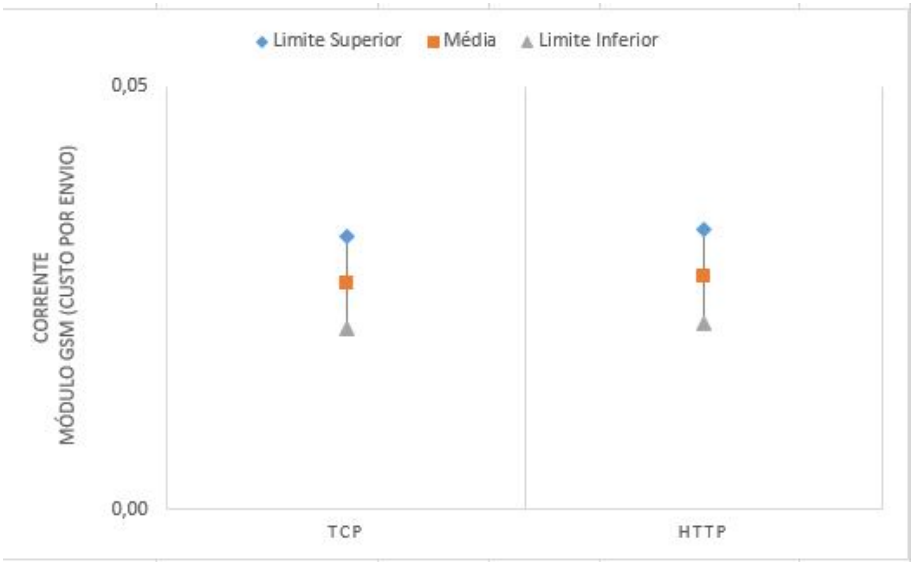
A média da corrente usando HTTP *Aplication* com intervalo de 1 minuto ficou em 0,026 A, Com corrente de 0,026 utilizando uma bateria que fornece ate 5Ah, a estimativa é que o sistema se mantenha funcionando por 192,30 horas, que dá em torno de 8,01 dias.

Já usando o TCP *Aplication*, para entregar a mesma quantidade de mensagens, a corrente média ficou em 0,027 A. Neste caso a estimativa é que o sistema se mantém funcionando por 185,1 horas, que dá em torno de 7,71 dias

Nas tabelas seguintes são apresentados resumos de todos os resultados obtidos, onde na Tabela 4 é apresentado a corrente médio, e os intervalos de confiança de cada modo durante todo o experimento, já na Tabela 5, é apresentado a média da corrente por mensagem enviada de acordo como intervalo e uma estimativa de quanto tempo o módulo se mantém funcionando.

A partir dos resultados obtidos, a forma de envio a ser escolhida para realizar o envio para o agente IoT deve ser de acordo com o tipo de cenário e aplicação, onde usando o modo HTTP *Aplication* obtivemos uma pequena maior autonomia em comparação com o TCP *Aplication*, porém o HTTP *Aplication* apresentou uma maior perda de pacotes em relação ao

Figura 14 – Média da corrente por envio de mensagem com intervalo de 1 minuto



Fonte: Elaborada pelo autor.

TCP *Aplication*, caso o ambiente monitorado seja sensível a perda de dados, a melhor opção é o TCP *Aplication*, caso o ambiente seja flexível a perda de dados, a melhor opção é o HTTP *Aplication* por apresentar um maior tempo de autonomia.

Tabela 5 – Resumo das Médias e Intervalos de Confiança

Intervalo	Modo de Envio	Métricas	Corrente(Ah)
Sem intervalo	HTPP Aplication	Limite Superior	0,050
		Média	0,044
		Limite Inferior	0,037
Sem intervalo	TCP Aplication	Limite Superior	0,051
		Média	0,045
		Limite Inferior	0,038
1 minuto	HTPP Aplication	Limite Superior	0,032
		Média	0,026
		Limite Inferior	0,021
1 minuto	TCP Aplication	Limite Superior	0,033
		Média	0,027
		Limite Inferior	0,021

Fonte: Elaborada pelo autor.

Tabela 6 – Consumo por Hora x Autonomia do Sistema

Intervalo	Modo de Envio	Consumo(Mensagem)	Autonomia(Hora)
Sem intervalo	HTPP Aplication	0,044 V	113,6
Sem intervalo	TCP Aplication	0,045 V	111,1
1 minuto	HTPP Aplication	0,026 V	192,30
1 minuto	TCP Aplication	0,027 V	185,18

Fonte: Elaborada pelo autor.

5.2.3 Tempo de Execução usando sensores virtuais

Este teste tem como objetivo identificar se a plataforma de *Middleware* Dojot consegue atender a uma grande quantidade de publicações sendo feitas ao mesmo tempo. Para realizar o teste de escalabilidade, primeiro foram criados os sensores e seus atributos na plataforma. Cada sensor possuía 2 atributos que eram valor sensor e temperatura. No sensor virtual foi criado uma função que gerava números aleatórios com intervalos de 1 à 100, esses sensores virtuais ficavam publicando na plataforma com intervalos de 10 segundos.

5.2.4 Execução do cenário de Escalabilidade

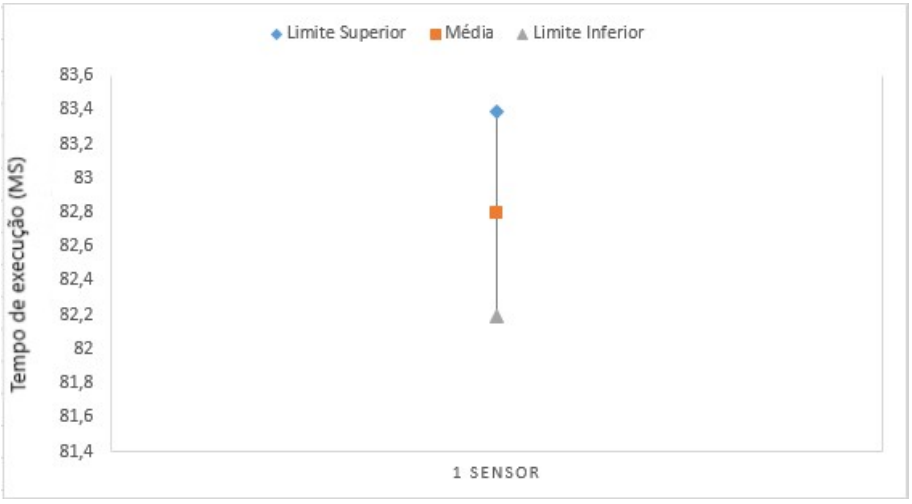
Para pegar o tempo de processamento que o sensor virtual leva pra gerar esses valores e publicar na plataforma, usamos de uma função do Java Script, conhecida como *console.time*, que nada mais é uma função que vai disparar a contagem do cronômetro, essa chamada é feita no inicio da função, já o *console.timeEnd* é feito no final da função, de modo que retorna o tempo de execução do sensor virtual. Cada teste foi repetido 10 vezes, e encerrado após a função retornar o tempo de execução.

5.2.5 Executando 1, 10 e 30 Sensores

Executando um único sensor após 10 repetições obtivemos uma média de tempo de processamento de 82,792 ms, com Limite Inferior de 82,190 e Superior de 83,393 ms. Executando 10 sensores após 10 repetições obtivemos uma média de tempo de processamento de 147,853 ms, com Limite Inferior de 118,219 e Superior de 177,486 ms. Executando 30 sensores após 10 repetições obtivemos uma média de tempo de processamento de 382,196 ms, com Limite Inferior de 343,763 e Superior de 420,629 ms, como podem ser vistos na Figura 17, 18 e 20. O Intervalo de confiança usado nesse experimento foi de 95%.

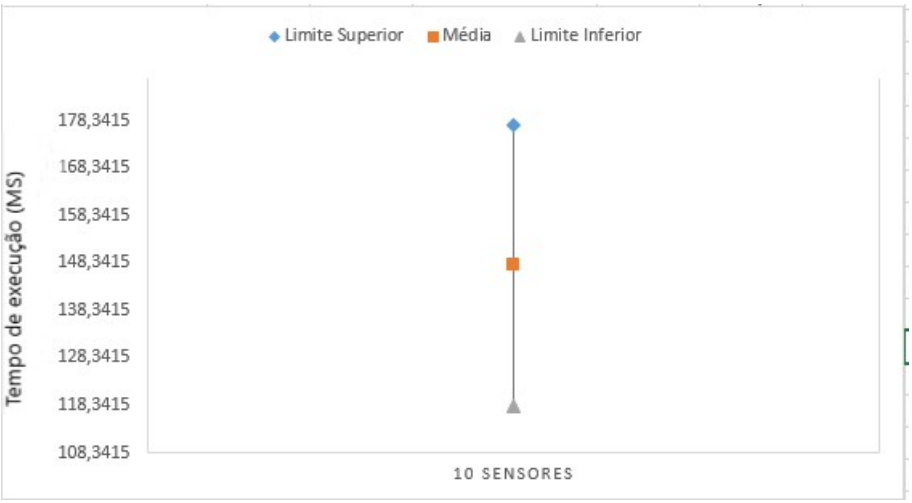
A seguir podemos observar de uma forma mais clara na Tabela 7 a quantidade de sensores usadas, a média e os Limites Inferiores e Superiores. Mesmo com 30 sensores virtuais enviando a cada 10 segundos, a plataforma apresentou um tempo de resposta aceitável.

Figura 15 – Média do tempo de execução



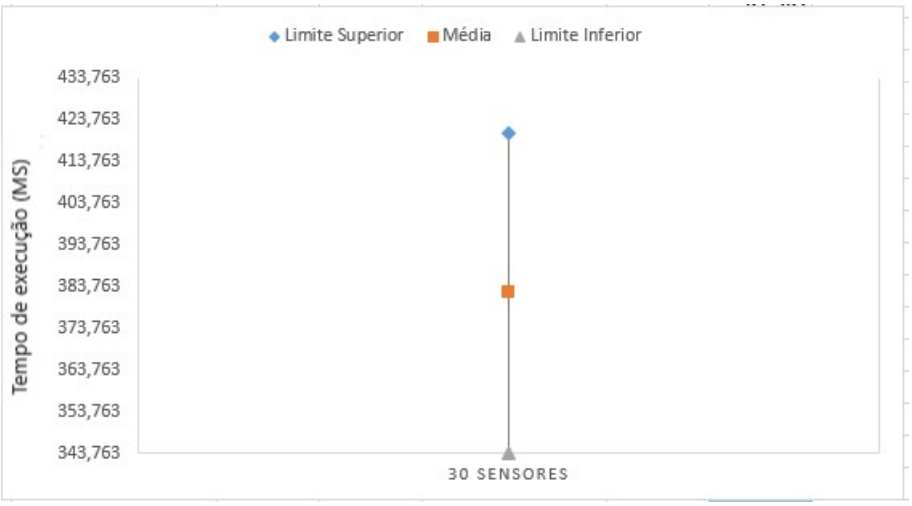
Fonte: Elaborada pelo autor

Figura 16 – Média do tempo de execução



Fonte: Elaborada pelo autor

Figura 17 – Média do tempo de execução



Fonte: Elaborada pelo autor

Tabela 7 – Médias e Intervalos de Confiança com o nº de Sensores

Nº de Sensores	Métricas A	Tempo (ms)
1	Limite Superior	83,393
1	Média	82,792
1	Limite Inferior	82,190
10	Limite Superior	177,486
10	Média	147,853
10	Limite Inferior	118,219
30	Limite Superior	420,629
30	Média	382,196
30	Limite Inferior	343,763

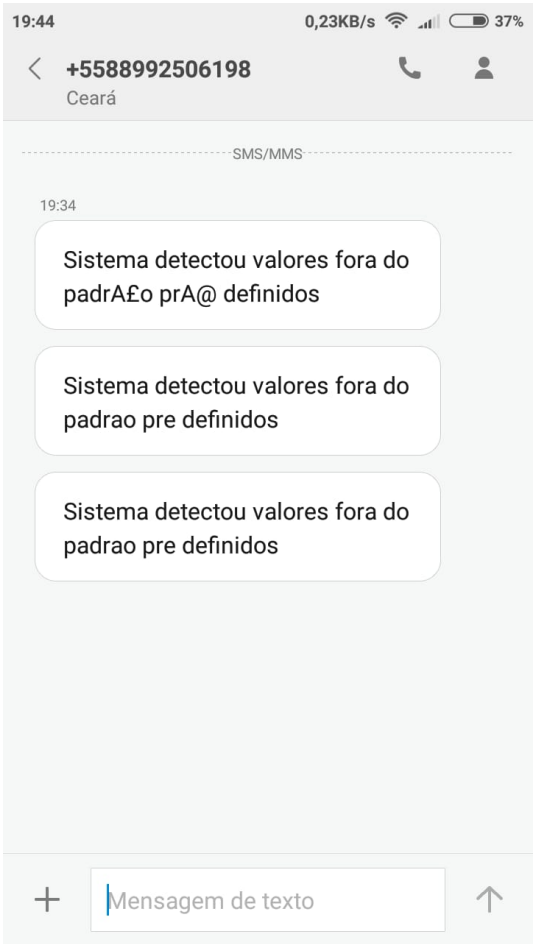
Fonte: Elaborada pelo autor.

6 SERVIÇO DE ALERTAS

A fim de alertar sobre possíveis variações fora do padrão pré-definido, esses valores tem relação ao Cenário que sera empregado, neste trabalho o valor pré-definido foi, caso a temperatura passe de 34 °C, envia um SMS.

Sendo assim foi adicionado ao código uma função de notificação, onde já temos um número telefônico previamente cadastrado para receber esses SMS, basicamente essa função compara o valor temperatura, que foi usado como exemplo, caso a temperatura apresenta valor maior que 34 °C, ele envia um SMS, após o envio do primeiro SMS, o segundo só é enviado após uma hora, podendo ser modificado esse intervalo. Na Figura 22 é apresentado um caso em que o envio foi realizado com sucesso. A primeira mensagem foi o teste para verificar se a função de envio estava funcionando. Já a segunda e terceira mensagem foi pra corrigir erros de acentuação e verificar se o intervalo de envio estava funcionando.

Figura 18 – Recebimento das notificações via SMS



Fonte: Elaborada pelo autor

Em comparação com os trabalhos relacionados, em apenas um foi abordado a questão do gasto energético dos módulo utilizados, no trabalho de (SILVA, 2016) voltado pra monitoramento de colmeias, foi feito o monitoramento da tensão dos nós sensores, onde após 9 horas a queda de tensão foi de 0,03 Volts, porém o *gateway* apresentou uma queda de 4,26 Volts durante esse mesmo período, onde a carga inicial da bateria era de 12,3 caindo para 7,69 Volts, encerrando o experimento já que não conseguia alimentar o *Gateway*, o problema dessa medição foi não ter monitorado a corrente que esse Gateway consumia, Embora o nosso experimento não tenha sido implementado em um Ambiente de monitoramento real, foi realizado tentando se aproximar o máximo possível da realidade, visto que foi feito em um local afastado, no qual não apresenta uma qualidade de sinal tão boa quanto na cidade, onde, no envio com intervalos de 1 minuto utilizando uma bateria de 5 Ah conseguimos uma autonomia de até 192,30 horas.

7 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou uma analise da Tecnologia GSM/GPRS aplicada em um Cenário de IoT, utilizando de uma plataforma IoT a fim de acelerar algumas fases do processo, como resultado obtivemos a criação de um sistema de monitoramento onde todos os valores coletados podem ser visualizados em tempo real através de um *Dashboard*.

Onde a analise foi feita a partir deste sistema criado, identificamos que o módulo GSM/GPRS apresentava duas formas de envio. onde no modo HTTP *Aplication* se apresentou um pequeno menor consumo que o modo TCP/IP *Aplication*. Em contrapartida, o modo TCP/IP Aplication *Aplication* apresentou uma taxa de perda de mensagens um pouco menor que o modo HTTP *Aplication*.

A partir das analises realizadas chegamos a conclusão que é uma tecnologia que pode ser usada em Cenarios de IoT, onde o modo de envio deve ser escolhido de acordo com o cenário que será monitorado, caso o ambiente de monitoramento seja flexível a perca de pacotes, a melhor opção é o HTTP *Aplication* por apresentar um maior tempo de autonomia. Caso o ambiente de monitoramento seja sensível a perca de pacotes, a melhor opção é o TCP/IP *Aplication* por apresentar uma maior taxa de entrega de pacotes, por fim o módulo GSM SIM800L se mostrou bastante viável e conseguiu suprir todas as necessidades deste trabalho.

Como trabalhos futuros, podemos citar a realização dos experimentos em um ambiental real, com sensores ligados diretamente ao Arduíno, além de realizar uma analise mais aprofundada do gasto energético. Outra etapa à ser realizada é a integração do banco diretamente com a plataforma, atualmente a plataforma se encontra na versão 2.0, onde um dos serviços disponíveis é a criação de alertas e notificação por e-mail, porém ainda não é uma versão estável, pois apresenta vários *bugs*, que impossibilitou a implantação desse serviço no sistema. Em trabalhos futuros pretende-se aproveitar o máximo dos recursos que essa plataforma disponibiliza.

Podemos afirmar que é possível realizar um sistema de monitoramento baseado em IoT através de matérias de baixo custo, que ajudam na identificação de problemas e auxiliam na tomada de decisões com autonomia de até 8 dias sem nenhuma intervenção.

Todos os códigos fontes embarcados no Arduíno, configurações e instalações necessárias que foram usados durante todo o processo de conclusão do sistema podem ser encontrados e baixados no Git Hub, a plataforma pode ser acessada através do IP 35.198.39.176:8000 com usuário *admin* e senha *admim*.

REFERÊNCIAS

ADVICE, E. **Unable to connect sim800l with network**. 2018. Disponível em: <http://www.embeddedadvice.com/t/unable-to-connect-sim800l-with-network/227>. Acesso em: 07 jun. 2018.

ALIPPI, C.; CAMPLANI, R.; GALPERTI, C.; ROVERI, M. A robust, adaptive, solar-powered wsn framework for aquatic environmental monitoring. **IEEE Sensors Journal**, IEEE, v. 11, n. 1, p. 45–55, 2011.

ALMEIDA, H. **Computação Brasil número 26**. 2015. Disponível em: <http://sbc.org.br/publicacoes-2/298-computacao-brasil>. Acesso em: 18 agosto. 2017.

ASHTON, K. That 'internet of things' thing. **RFiD Journal**, v. 22, n. 7, p. 97–114, 2009. Acesso em: 01 jan. 2017.

BANDYOPADHYAY, S.; SENGUPTA, M.; MAITI, S.; DUTTA, S. Role of middleware for internet of things: A study. **International Journal of Computer Science and Engineering Survey**, Academy & Industry Research Collaboration Center(AIRCC), v. 2, n. 3, p. 94–105, 2011.

DARNELL, L. **The Internet of Things**: a look at real-world use cases and concerns. [S.l.]: Kindle Edition, 2015. v. 1.

DOJOT. **Dojot Soluções para IoT**. 2018. Disponível em: <http://www.dojot.com.br/sobre-a-dojot-iot/>. Acesso em: 20 Dez. 2017.

FIWARE. **protocolo MQTT como alternativa para comunicaca**

o IoT. 2016. Disponível em: <http://fiwarelabsp.org/2016/07/apresentacao-do-protocolo-mqtt/>. Acesso em: 20 jan. 2018.

FRAMINGHAM, M. **IDC Publishes Three Landmark Reports in the IoT Space**. 2015. Disponível em: <http://www.idc.com/getdoc.jsp?containerId=prUS25658015>. Acesso em: 18 fev. 2018.

GEORGIEV, T.; GEORGIEVA, E.; SMRIKAROV, A. M-learning-a new stage of -learning. In: **International conference on computer systems and technologies-CompSysTech**. [S.l.: s.n.], 2004. v. 4, n. 28, p. 1–4.

HUNKELER, U.; TRUONG, H. L.; STANFORD-CLARK, A. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In: IEEE. **Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on**. [S.l.], 2008. p. 791–798.

KRIDI, D. S. **Monitoramento de padrões térmicos em colmeias de abelhas via redes de sensores sem fio**. 2014. 62 f. Dissertação (mestrado)- Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza-CE, 2014.

L; FILHO, R. M. **Sistema de monitoramento inteligente de uma horta escolar baseado na plataforma arduino**, Universidade do Estado do Amazonas, 2018.

LACERDA, F. **Arquitetura da Informacao Pervasiva: projetos de ecossistemas de informacao na Internet das Coisas**. [S.l.: s.n.], 2016.

LEDESMA, N. E. C. **Desenvolvimento de um sistema de SHM sem fio e com compensação automática de temperatura**, Universidade Estadual Paulista (UNESP), 2015. Disponível em: <http://hdl.handle.net/11449/132167>. Acesso em: 18 out. 2017.

MA, J.; ZHOU, X.; LI, S.; LI, Z. Connecting agriculture to the internet of things through sensor networks. In: IEEE. **Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing**. [S.l.], 2011. p. 184–187.

MARIOTE, L. **Plataforma aberta para desenvolvimento de IoT Dojot**. [S.l.: s.n.], 2017.

MOTA, J. **Como enviar SMS e fazer chamadas com Arduino**. 2017. Disponível em: <https://www.arduinoportugal.pt/enviar-sms-arduino-sim800l>. Acesso em: 14 jan. 2018.

MOULY, M.; PAUTET, M.-B.; BY-HAUG, T. F. **The GSM system for mobile communications**. [S.l.]: Telecom publishing, 1992.

NAUMOWICZ, T. *et al.* Wireless sensor network for habitat monitoring on skomer island. In: IEEE. **Local Computer Networks (LCN), 2010 IEEE 35th Conference on**. [S.l.], 2010. p. 882–889.

NICOLAU, A. **CPqD lanca dojot, plataforma aberta para desenvolvimento de aplicac**

oes de IoT. 2018. Disponível em: <https://www.itforum365.com.br/tecnologia/cpqd-lanca-dojot-plataforma-aberta-para-o-desenvolvimento-de-aplicacoes-de-iot/>. Acesso em: 18 jan. 2018.

OLIVEIRA, L. M.; RODRIGUES, J. J. Wireless sensor networks: a survey on environmental monitoring. **JCM**, v. 6, n. 2, p. 143–151, 2011.

OLIVEIRA, T. R.; GIGLIO, G. P. de M. Análise de estudo de casos em abordagens pelo mundo da implementação de internet das coisas. **Caderno de Estudos em Sistemas de Informação**, v. 1, n. 2, 2018.

PETTS, J. **Handbook of Environmental Impact Assessment: Volume 2: impact and limitations**. [S.l.]: John Wiley & Sons, 2009. v. 2.

PIRES, P. F. *et al.* Plataformas para a internet das coisas. **Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**, 2015.

RAO, B. P. *et al.* Cloud computing for internet of things & sensing based applications. In: IEEE. **Sensing Technology (ICST), 2012 Sixth International Conference on**. [S.l.], 2012. p. 374–380.

SAHÃO, S. J. **Plataforma aberta para desenvolvimento de IoT**. [S.l.: s.n.], 2017.

SILVA, A. d. L. a. **Monitoramento não invasivo de colmeias através da IOT**, 2016. Disponível em: <http://www.repositoriobib.ufc.br/00003a/00003ac0.pdf>. Acesso em: 15 fev. 2017.

SÔNEGO, A. A. *et al.* **A Internet das Coisas aplicada ao conceito de eficiência energética**. [S.l.: s.n.], 2017.

TATEOKI, G. T. *et al.* **Monitoramento de dados via internet baseado em telefonia celular**. 2007. São Paulo: Universidade Estadual Paulista (UNESP).

YUAN, M. **Conhecendo o MQTT**. 2017. Disponível em: <https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html>. Acesso em: 12 mar. 2018.

APÊNDICE A – CÓDIGOS-FONTE DA COMUNICAÇÃO COM O AGENTE IOT

Código-fonte 1 – Modo de Envio HTTP Aplication

```
1
2  #include <SoftwareSerial.h>
3  #include <JeeLib.h>
4  #include <gprs.h>
5  ISR(WDT_vect) {
6    Sleepy::watchdogEvent();
7  }
8
9  SoftwareSerial SIM800L(10, 11); // Configuraç o  da Porta Serial
10
11 String virgula = ",";
12 String meio = "&temperatura=";
13 int valorsensor = 35;
14 float temperatura = 37.2;
15
16 void resposta() {
17   while (SIM800L.available())
18   {
19     if (SIM800L.available() > 0)   //recebe os dados
20     {
21       Serial.write(SIM800L.read()); //impreme a resposta
22
23     }
24   }
25
26   Serial.flush();
27   Serial.flush();
28 }
29
30 void setup() {
31   // put your setup code here, to run once:
```

```
32    Serial.begin(9600); //iniciando comunica    £o serial
33    SIM800L.begin(9600); //iniciando SoftwareSerial
34
35    SIM800L.println("AT"); //enviando Comando AT
36    delay (1000); //Tempo de espera
37    resposta();
38
39    //  GetSignalQuality();
40  }
41
42  void loop() {
43
44    gsm_enviohttp();
45    delay (1000);
46    // GetBattery();
47    //SIM800L.println("AT+CSCLK=1");
48    //delay (1000); //Tempo de espera
49    //resposta();
50
51    // for (byte i = 0; i < 1; ++i) Intervalo de envio é definido
         nesse for
52    // Sleepy::loseSomeTime(60000);
53
54  }
55
56  void gsm_enviohttp() {
57
58    SIM800L.println("AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\""); //enviando
         Comando AT
59    delay (1000); //Tempo de espera
60    resposta();
61
62    SIM800L.println("AT+SAPBR=3,1,\"APN\",\"claro.com.br\""); //
         enviando Comando AT
```

```
63  delay (1000); //Tempo de espera
64  resposta();
65
66  SIM800L.println("AT+SAPBR=3,1,\"USER\",\"claro\""); //enviando
       Comando AT
67  delay (1000); //Tempo de espera
68  resposta();
69
70  SIM800L.println("AT+SAPBR=3,1,\"PWD\",\"claro\""); //enviando
       Comando AT
71  delay (1000); //Tempo de espera
72  resposta();
73
74  SIM800L.println("AT+SAPBR=1,1");
75  delay (2000);
76  resposta();
77
78  SIM800L.println("AT+SAPBR=2,1");
79  delay (2000);
80  resposta();
81
82  SIM800L.println("AT+HTTPINIT"); //INICIALIZANDO SERVIÇO HTTP
83  delay (2000);
84  resposta();
85
86  SIM800L.println("AT+HTTPPARA=\"CID\",1");
87  delay (2000);
88  resposta();
89
90  //CONCATENANDO OS VALORES PARA REALIZAR O ENVIO
91  String finali = "\"";
92  String mensagem = "AT+HTTPPARA=\"URL\",\"http
       ://200.129.39.91:3000/?valorsensor=";
```

```
93    String concatena = String(mensagem) + String (valorsensor) +
          String(meio) + String(temperatura) + String(finali);
94
95    SIM800L.println(concatena);
96    delay (2000);
97    resposta();
98
99    SIM800L.println("AT+HTTPACTION=0");
100   delay (3000);
101   resposta();
102
103   SIM800L.println("AT+HTTPREAD");
104   delay (3000);
105   resposta();
106
107   SIM800L.println("AT+HTTPTERM");
108   delay (1000);
109   resposta();
110
111   SIM800L.println("AT+SAPBR= 0,1"); //EHCERRA
112   delay (2000);
113   resposta();
114
115   //
116   delay(1000);
117   GetBattery();
118
119 }
120
121
122 void GetBattery () {
123   String response = "";
124   long int time = millis();
125   Serial.println("Porcentagem da Bateria!");
```

```
126   SIM800L.println("AT+CBC");
127   while ( (time + 3000) > millis()) {
128     while (SIM800L.available()) {
129       response += char(SIM800L.read());
130     }
131   }
132   Serial.print(response);
133 }
```

Código-fonte 2 – Modo de Envio TCP/IP Aplication Aplication

```
1
2  #include <gprs.h>
3  #include <JeeLib.h> // https://github.com/jcw/jeelib
4  ISR(WDT_vect) { Sleepy::watchdogEvent(); } // interrupt handler for
       JeeLabs Sleepy power saving
5
6  //MUDANÇA PARA RECEBER OS VALORES DOS SENSORES EM UMA STRING :D
7  #include <SoftwareSerial.h>
8  #include <String.h>
9
10 SoftwareSerial mySerial(10, 11);
11
12 int valorsensor = 36;
13 String meio = "&temperatura=";
14 String espaco = " ";
15 float temperatura = 37.8;
16 String Z = "HTTP/1.1\r\nHost: 200.129.39.91\r\nConnection: close\r\
       n\r\n";
17
18 void setup()
19 {
20   mySerial.begin(9600);               // the GPRS baud rate
21   Serial.begin(9600);    // the GPRS baud rate
```

```
22    delay(1000);
23  }
24
25  void loop()
26  {
27    //    temp=analogRead(A0);
28   // temp = temp * 0.4887;
29
30    delay(2000);
31    Send2Pachube();
32   // mySerial.println("AT+CSCLK=2");
33  //  for (byte i = 0; i < 5; ++i)
34  //  Sleepy::loseSomeTime(60000);
35  //{
36  //  Sleepy::loseSomeTime(60000);
37  //  }
38    if (mySerial.available())
39      Serial.write(mySerial.read());
40  }
41  void Send2Pachube()
42  {
43    mySerial.println("AT+CBC");
44    delay(1000);
45
46    mySerial.println("AT+CPIN?");
47    delay(1000);
48
49    mySerial.println("AT+CREG?");
50    delay(1000);
51
52    mySerial.println("AT+CGATT?");
53    delay(1000);
54
55    mySerial.println("AT+CIPSHUT");
```

```
56    delay(1000);
57
58    mySerial.println("AT+CIPSTATUS");
59    ShowSerialData();
60    delay(2000);
61
62    mySerial.println("AT+CIPMUX=0");
63    delay(2000);
64
65    ShowSerialData();
66
67    mySerial.println("AT+CSTT=\"claro.com.br\",\"claro\",\"claro\"");
         //start task and setting the APN,
68    delay(1000);
69
70    ShowSerialData();
71
72    mySerial.println("AT+CIICR");//bring up wireless connection
73    delay(3000);
74
75    ShowSerialData();
76
77    mySerial.println("AT+CIFSR");//get local IP adress
78    delay(2000);
79
80    ShowSerialData();
81
82    mySerial.println("AT+CIPSPRT=0");
83    delay(3000);
84
85    ShowSerialData();
86
87    mySerial.println("AT+CIPSTART=\"TCP\",\"200.129.39.91\",\"3000\""
         );//start up the connection
```

```
88
89    delay(3000);
90
91    ShowSerialData();
92    delay(3000);
93    mySerial.println("AT+CIPSEND");//begin send data to remote server
94
95    delay(1000);
96    ShowSerialData();
97
98
99    String inicio = "GET http://200.129.39.91:3000/?valorsensor=" +
          String(valorsensor) + String (meio) + String(temperatura) +
          String (espaco);
100   ShowSerialData();
101   delay(100);
102   mySerial.print(inicio);
103   mySerial.print(Z);
104   delay(3000);
105   ShowSerialData();
106
107   mySerial.println((char)26);//sending
108   delay(5000);//waitting for reply, important! the time is base on
          the condition of internet
109   mySerial.println();
110
111   ShowSerialData();
112
113   mySerial.println("AT+CIPSHUT");//close the connection
114   delay(1000);
115   ShowSerialData();
116 }
117 void ShowSerialData()
118 {
```

```
119   while (mySerial.available() != 0)
120     Serial.write(mySerial.read());
121 }
```

APÊNDICE B – FUNÇÃO RESPONSÁVEL PELO ENVIO DE SMS DE ALERTA

Código-fonte 3 – Função que realiza alertas via SMS caso os valores monitorados saiam do padrões pré definidos

```
1  void sms (){
2   int var = 0;
3   while (var == 0 && temperatura > 33){
4  //  if (temperatura > 33){
5
6    SIM800L.write("AT+CBC");
7    delay(1000);
8
9    SIM800L.write("AT+CMGF=1\r\n");
10   delay(1000);
11
12   SIM800L.write("AT+CMGS=\"88993584050\"\r\n");
13   delay(1000);
14
15   SIM800L.write("Sistema detectou valores fora do padrao pre
        definidos");
16   delay(1000);
17
18   SIM800L.write((char)26);
19   delay(1000);
20   var++;
21   }
22   if (var >= 1 && temperatura > 33)
23   {
24   for (byte i = 0; i < 60; ++i) //Intervalo de envio é definido
        nesse for
25   Sleepy::loseSomeTime(60000);
26   SIM800L.write("AT+CMGF=1\r\n");
27   delay(1000);
28
```

```
29    SIM800L.write("AT+CMGS=\"88993584050\"\r\n");
30    delay(1000);
31
32    SIM800L.write("Sistema detectou valores fora do padrao pre
          definidos");
33    delay(1000);
34
35    SIM800L.write((char)26);
36    delay(1000);
37
38      } else {
39        Serial.println("Temperatura normal");
40        }
41
42    }
```