PROPOSTA DE PADRONIZAÇÃO DE COMUNICAÇÃO PARA DISPOSITIVOS IOT

Daniel V. A. Silveira*, Josiel Patrício*, Judson S. Costa*, Diego R. C. Silva*, Marcelo B. Nogueira*, Marconi C. Rodrigues*

*Escola de Ciências e Tecnologia / UFRN
Campus Universitário, Lagoa Nova
Natal, RN, Brasil

Emails: danielaraujo@bct.ect.ufrn.br, patricio@bct.ect.ufrn.br, judson.scosta@bct.ect.ufrn.br, diego@ect.ufrn.br, marcelonogueira@ect.ufrn.br, marconicamara@ect.ufrn.br

Abstract— This article proposes a semantic standardization for intelligent devices that are in the context of the Internet of things. The proposed format abstracts low level logical concepts so that when used in the application layer, it has an easy adaptation and interoperability with different services. The standard proposes to describe any generic device based on a structure of sensors and controllers, so it can describe from a lamp to a car, opening the possibility for an interaction between hardware services, cloud for data storage, applications capable of generating statistics, cross-platform device monitoring, and end-user control interfaces.

Keywords— Automation system, Internet of Things, standardization, application.

Resumo— O presente artigo propõe uma padronização semântica para dispositivos inteligentes que se encontram no contexto de Internet das coisas. O formato proposto abstrai conceitos de baixo nível lógico para que quando usados na camada de aplicação, possua uma fácil adaptação e interoperabilidade com serviços distintos. O padrão propõe descrever qualquer dispositivo genérico se baseando em uma estrutura de sensores e controladores, podendo assim descrever desde uma lâmpada até um automóvel, abrindo a possibilidade para uma interação entre serviços de *hardware*, nuvem para armazenamento de dados, aplicações capazes de gerar estatísticas, monitoramento de dispositivos multiplataforma, e interfaces de controle para usuários finais.

Palavras-chave— Sistema de automação, Internet das coisas, padronização, aplicação.

1 Introdução

A indústria passa pela sua quarta revolução, a qual está inserida no contexto de Internet das coisas do inglês "Internet of Things" (IoT). Embora sejam encontradas diversas definições de IoT na literatura, será utilizada a definição de Luigi Atzori na qual IoT é um novo paradigma cuja ideia básica dentro deste conceito é a presença generalizada de uma variedade de coisas ou objetos - tais como etiquetas RFID, sensores, atuadores, telefones celulares, entre outros - que, através de esquemas únicos de endereçamento, são capazes de interagir uns com os outros e cooperar com seus vizinhos para alcançar objetivos comuns (Atzori et al., 2010). Para possibilitar essa interação e cooperação é necessário conectividade entre os dispositivos e uma devida interoperabilidade entre as coisas. Diversas soluções ao prolema da interoperabilidade na IoT foram propostas nas diversas camadas da pilha de protocolos de Internet, conforme demonstrado na metodologia, porém essas propostas não são aplicáveis a todos os casos, a depender de cada projeto, as necessidades de comunicação divergem. Nesse cenário de diversidade de protocolos e padrões de comunicação é importante uma padronização no conteúdo das mensagens trocadas entre os dispositivos e interfaces de controle e visualização, sem que seja necessária uma revisão na escolha dos protocolos adotados ou mesmo nas tecnologias utilizadas. Com a utilização de mensagens padronizadas, é possível coletar e processar dados e intervir sobre as variáveis controladas por meio desses aparelhos possibilitando uma melhor adaptação desses produtos e do sistema como um todo, otimizando o desempenho e a eficiência, reduzindo desperdícios, aumentando a confiabilidade, segurança e conforto para o caso dos usuários.

Há diversos dispositivos de IoT no mercado que são para uso específicos e de padrões fechados, alguns inclusive não possuem uma padronização para comunicação entre si. Caso haja uma comunicação, é feita por meio de um *gateway*, como os produtos da Philips® que possuem padrões fechados, por exemplo, e, em grande parte dos casos os serviços não aceitam dispositivos de outras empresas. Diversas vezes a incompatibilidade é remediada com um desenvolvimento específico de um *middleware* para lidar com as diferentes arquiteturas, voltado para que produtos de empresas diferentes utilizem e forneçam serviços compatíveis. Para Wagner Macêdo, a interligação fim-a-fim é o recurso que indica que a solução permite que nós clientes e nós servidores interajam de forma espontânea, isto é, sem que estejam cientes de que estão se comunicando através de um intermediário (Macêdo et al., 2016) e inclusive propõe em seu trabalho uma arquitetura de *gateway* de Camada de Aplicação.

O presente artigo propõe uma padronização semântica genérica para dispositivos de Internet

das coisas, permitindo uma comunicação e intercâmbio de serviços entre dispositivos sem compatibilidade prévia entre os equipamentos. O padrão proposto visa permitir a descrição de qualquer maquinário em atributos para facilitar o desenvolvimento sem que seja necessário alteração em nenhuma das camadas escolhidas previamente pelos desenvolvedores, beneficiando o ambiente de produção de novas ferramentas inteligentes por empresas ou grupos de pesquisa, bem como permitindo a integração de diversos dispositivos sejam eles compostos por um conjunto de sensores, de atuadores ou de ambos os fins e independente de quais protocolos de comunicação utilizem.

O trabalho está organizado de forma a apresentar a pilha de protocolos da Internet com as tecnologias e padrões mais usados em cada camada na seção de Revisão de literaturada e, posteriormente, descrever como foi desenvolvido o padrão de troca de mensagens proposto na seção Metodologia. Trazer, em outra seção, discussões sobre a proposta e implementação. Apresentar ao leitor os resultados obtidos com a utilização do padrão proposto. E, numa última seção, apresentar propostas para trabalhos futuros, a fim de deixar a pesquisa mais completa e o padrão mais genérico e robusto quanto possível.

2 Revisão de literatura

Atualmente é indiscutível o domínio das tecnologias de comunicação sem fio, a quantidade e velocidade com que novas tecnologias vêm sendo desenvolvidas para esse fim só mostram quão diversificado o mundo IoT está ficando. Um pequeno exemplo da quantidade de formas de conexão existente pode ser visto tomando como exemplo alguns modelos de aparelhos celulares que possuem conectividade por diversas tecnologias como *global system for mobile communication (GSM), High Speed Packet Access (HSPA) evolution (HSPA+), long term evolution (LTE), Near Field Communication (NFC), IEEE 802.11 (WI-FI®), IEEE 802.15.4 (Bluetooth)*, entre outras, tudo está presente em um único aparelho. Wi-Fi® é uma marca registrada da empresa *Wi-Fi Alliance*. Alguns padrões já utilizados em IoT serão mencionados a seguir utilizando-se como referência a pilha de protocolos da Internet formada por cinco camadas: física, de enlace, de rede, de transporte e de aplicação (Kurose, 2013). Essa organização tem por objetivo melhorar a apresentação e facilitar a compreensão de quais níveis cada padrão se encaixa.

O IEEE 802 é um grupo de normas visando padronizar redes locais e metropolitanas nas camadas física e de enlace por tanto as padronizações dessas camadas geralmente iniciam com IEEE 802. Nessas camadas mais baixas da pilha de protocolos, física e de enlace, dentre os padrões utilizados em IoT é possível citar o padrão IEEE 802.3 - *Ethernet*, tecnologia dominante de comunicações por cabo com quatro diferentes taxas de dados suportadas usando cabos não blindados de pares trançados - *Unshielded Twisted Pair* (UTP) (Christensen et al., 2010). Nele são especificados a forma de cabeamento e também a sintaxe e semântica *Média Access Control* (MAC) da camada de enlace. Esse padrão ainda é usado em projetos atuais o desenvolvido no artigo (Acciari et al., 2018).

Dentre os padrões que não utilizam cabeamento (*wireless*), são encontrados os padrões IEEE 802.11 (Wi-Fi), também implementa e descreve o protocolo MAC na camada de enlace, e IEEE 802.15.4, este último dando origem a algumas tecnologias desenvolvidas especificamente para IoT. O *Wi-Fi* se destaca pela facilidade de uso e pela quantidade de dispositivos capazes de se conectar nativamente a esse padrão, bem como a facilidade de acesso, uma vez que no Brasil muitas empresas que proveem acesso à Internet instalam modem com roteador Wi-Fi integrado. A facilidade de acesso, a grande quantidade de dados que podem ser enviados, a velocidade e a latência nas comunicações ainda não tornam a tecnologia Wi-Fi a dominante no segmento IoT pelo seu consumo de energia, embora proposta de baixo consumo (*low-power Wi-Fi*) tenha sido testada (Tozlu et al., 2012), tecnologias baseadas no IEEE 802.15.4 ainda apresentam melhores resultados em baixo consumo de energia. Há no IEEE o grupo 802.11ba *Battery Life Improvement* focado em um padrão de Wi-Fi para dispositivos IoT de baixo consumo alimentados por bateria (McCormick, 2017).

Em contrapartida, as tecnologias baseadas no padrão IEEE 802.15.4 apresentam menores taxas de transmissão, menor largura de banda, porém um consumo menor de energia o que as torna mais indicadas para IoT quando aplicada a dispositivos de baixo consumo de energia (*low-power ou ultra low-power*)(Augustin et al., 2016). O IEEE 802.15.4 especifica a camada física e a camada de enlace de dados para redes de área pessoal, possui suporte a três bandas de frequências, 868 MHz, 928 MHz e 2,4 GHz e pode oferecer taxas de dados de até 250 kbit/s a depender das condições do ambiente. Baseado nesse padrão em 1999 foi lançado o *Bluetooth 1.0* com raio de transmissão utilizado na prática de 10m. Atualmente está na versão 4.0 e conta com uma versão *Low Energy* (BLE) que consome realmente pouca energia e alcança taxas de transmissão de 1Mbit/s com alcance de até 50m. Os dispositivos BLE podem operar no papel mestre ou escravo. Um mestre pode gerenciar múltiplas conexões simultâneas com vários dispositivos escravos, mas um escravo só pode ser conectado a um único mestre. Portanto, uma topologia de rede BLE é uma estrela. Diferentemente

do *Bluetooth* comum, a descoberta é feita para que o escravo anuncie em um ou vários dos três canais de propagação designados. O mestre escaneia esses canais para descobrir escravos. Após a descoberta, a transmissão de dados acontece na forma de eventos de conexão em que o mestre e o escravo acordam em sincronia para trocar quadros. Ambos os dispositivos entram em modo de economia de energia (Siekkinen et al., 2012). O *ZigBee*™ atualmente mantido pela *ZigBee Alliance* opera conforme o IEEE 802.15.4, quando na frequência de 2.4 GHz pode sofrer interferência de redes IEEE 802.11 (Petrova et al., 2007).

O LoRa cujo nome vem do termo em inglês *Long Range*, isto é, longo alcance, como o nome sugere, é um sistema de comunicação sem fio de longo alcance, promovido pela LoRa *Alliance*. Este sistema visa ser utilizável em dispositivos alimentados por baterias de longa duração, onde o consumo de energia é de suma importância. LoRa pode comumente se referir a duas camadas distintas: (i) uma camada física usando a técnica de modulação de rádio *Chirp Spread Spectrum* (CSS); e (ii) um protocolo de camadas superiores chamado LoRaWAN, embora o sistema de comunicação LoRa também implique uma arquitetura de rede de acesso específica. A camada física LoRa, desenvolvida pela Semtech, permite comunicações de longo alcance, baixa potência e baixo rendimento. Ele opera nas bandas ISM de 433, 868 ou 915 MHz, dependendo da região em que é implementado. A carga útil de cada transmissão pode variar de 2 a 255 octetos, e a taxa de dados pode alcançar até 50 Kbps quando a agregação de canal é empregada. A técnica de modulação é uma tecnologia proprietária da Semtech. O LoRaWAN fornece um mecanismo de controle de acesso médio, permitindo que muitos dispositivos finais se comuniquem com um *gateway* usando a modulação LoRa. Enquanto a modulação LoRa é proprietária, o LoRaWAN é um padrão aberto que está sendo desenvolvido pela LoRa *Alliance*. Esses padrões são usados entre os dispositivos e os *gateways*. A partir desses *gateways* ocorre a conexão com servidores via protocolo IP padrão (Centenaro et al., 2016).

Existe ainda o SigFox, usa modulação de banda ultra estreita, opera em 200 kHz de banda disponível ao público para trocar mensagens de rádio pelo ar. Cada mensagem tem 100 Hz de largura e transfere a 100 ou 600 bits/s, dependendo da região. Assim, longas distâncias podem ser alcançadas enquanto são muito robustas contra o ruído. Uma mensagem *uplink* tem uma carga útil de até 12 bytes e leva uma média de 2s pelo ar para alcançar as estações de base que monitoram o espectro procurando por sinais UNB(Banda ultra estreita) para demodular. Para uma carga de dados de 12 bytes, um quadro Sigfox usará 26 bytes no total. A permissão de carga útil em mensagens *downlink*https://www.overleaf.com/ é de 8 bytes (Sigfox S.A., 2018).

Na **Camada de transporte** os protocolos TCP (*Transmission Control Protocol*) e UDP (*User Datagram Protocol*) são responsáveis pelo transporte de dados entre máquinas independente da aplicação e topologia da rede. O protocolo TCP oferece confirmação de recebimento da chegada dos dados, possibilitando a certeza da conexão e comunicação. Já no protocolo UDP não há confirmação, acelerando o envio e removendo a verificação, por apresentar maior velocidade é usado em *streaming* de dados como vídeos e músicas.

No topo da pilha há a **Camada de aplicação** com os protocolos de comunicação para uma interligação fim a fim entre aplicações. Existem diversos protocolos com finalidades distintas para diversos contextos, enquanto alguns protocolos apresentam suporte à aplicações genéricas, outros têm como finalidade solucionar problemas específicos. Como exemplo pode ser utilizado o *Hypertext Transfer Protocol* (HTTP) um protocolo criado em 1990 amplamente utilizado para a transferência de textos estruturados (Fielding et al., 1999). Tem como padrão de troca de mensagens o formato pedido/resposta. No entanto, para a utilização em situações que se tem um grande número de requisições e aplicações em tempo-real, como em alguns contextos de IoT e *Machine to Machine* (M2M), é gerado uma sobrecarga na aplicação (Yokotani and Sasaki, 2016).

É nessas situações que o *Message Queuing Telemetry Transport* (MQTT) apresenta um melhor desempenho e tem como padrão de troca de mensagens o formato publicar-assinar que permite a distribuição de mensagens um-para-muitos (Standard, 2014).

3 Metodologia da pesquisa

Primeiramente foi realizada uma pesquisa do estado da arte da IoT, depois disso foi criada uma instância de uma arquitetura IoT a fim de realizar os testes do padrão proposto. Para os testes nos dispositivos buscou-se um microcontrolador com capacidade de comunicação por redes sem fio integrada IEEE 802.11 para utilização das redes já disponibilizadas pela Universidade e também pela necessidade de troca de dados acentuada, como mencionado em seções anteriores, para monitoramento constante de algumas variáveis, bem como o fato de não precisar da aquisição de hardware adicional e fácil integração com a rede e os sistemas já desenvolvidos e disponíveis no laboratório em que os testes foram realizado. Após a escolha do MCU para compôr os dispositivos de teste, foram adquiridos sensores e atuadores e posteriormente desenvolvido um sistema para a realização dos testes de conectividade, desempenho e facilidade de implementação de protocolos de comuni-

cação tanto na parte do servidor como também nos dispositivos.

A instância da arquitetura IoT na qual foi testado e validado o padrão proposto é descrita a seguir e conta com uma aplicação web para automação, sensores de temperatura e de movimento, de consumo de água (hidrômetros), lâmpadas com controle de intensidade e cor.

3.1 Ambiente de testes

O ambiente de testes é, como demonstrado na Figura 1, composto por uma aplicação web (*client-side*) que contém a interface tanto para visualização dos dados provenientes dos sensores quanto para fazer o controle dos atuadores vinculados a um dos dispositivos que estão cadastrados no sistema e uma aplicação servidora (*server-side*) que armazena todos os dados pertinentes aos dispositivos.

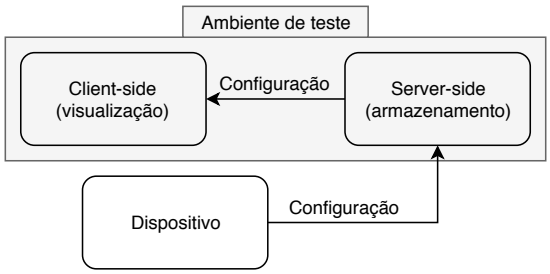

Figura 1: Ambiente de teste

3.1.1 *Client-side*

A aplicação web foi implementada utilizando o *framework React.js* que auxilia o desenvolvimento de interfaces gráficas. Por meio de uma interface, o usuário gerencia e interage com os dispositivos cadastrados. Para ter uma interação, primeiro a interface faz uma requisição a uma aplicação *server-side* que retorna todos os dados dos dispositivos no formato que está sendo proposto e será explicado na seção 3.2, assim, identifica-se os sensores e controladores que estão contidos em cada dispositivo e adapta-se o formato de visualização compatível com cada um desses equipamentos. Isso torna essa aplicação dinâmica para a interação de quaisquer controladores e visualização de quaisquer sensores de um dispositivo.

Uma característica importante é a possibilidade da criação de quaisquer tipos de sensores e atuadores que constituem um dispositivo e isso ser compatível com uma interface qualquer que tenha suporte a esse novo tipo. Um exemplo pode ser a criação de um controlador que tem como interface de controle um volante com uma rotação de 270° graus para os lados direito e esquerdo e um passo de 5°. Uma aplicação web receber a configuração de um dispositivo com controlador(es) desse formato e montar a interface de controle de acordo com as especificações definidas. Dessa forma é necessário o envolvimento apenas do desenvolvedor do *hardware* que precisa de um novo controlador para um dispositivo desenvolvido e do criador da interface web para dar suporte a essa nova forma de controlador.

3.1.2 *Server-side*

A aplicação servidora foi implementada utilizando o *framework Node.js* que fornece fluxo assíncrono, leitura e gravação de dados de forma não bloqueante e diversos outros recursos para a construção de aplicações. Ela foi feita com o intuito de centralizar as tomadas de decisão e participar da arquitetura como (*middleware*).

O *middleware* é compatível com os protocolos de comunicação HTTP, *WebSocket* e MQTT. Isso permite a utilização de dispositivos com diferentes finalidades e que podem ser suportados por um dos protocolos disponíveis, satisfazendo as necessidades de cada contexto em que um dispositivo está inserido, como aplicações em tempo-real ou simples reportes de um determinado sensoriamento.

O formato do envio de dados escolhido foi o *JavaScript Object Notation (JSON)* por ser um formato condensado para transferência de dados que não apresenta uma notação verbosa como formatos existentes. Atualmente, o JSON é suportado por diversas linguagens de programação (Crockford, 2006).

3.2 Estrutura de cadastro

A estrutura demonstrada na Listagem 1, criada pela padronização proposta no presente artigo, permite que um desenvolvedor defina um produto a partir de um JSON de configuração que possui detalhes sobre os dispositivos e o funcionamento de sensores e atuadores vinculados. A formatação da Listagem 1 define um produto com apenas um dispositivo composto por sensores e controladores:

```
[
  {
    "name": "nome do dispositivo",
    "serial": "numero_de_serie",
    "ip": "192.168.0.27",
    "protocol": "ws",
    "sensors": [],
    "controllers": []
  }
]
```

Listing 1: JSON de cadastro

3.2.1 Controlador

É definido como a menor estrutura de controle em um dispositivo, um mesmo controlador pode comandar um ou mais atuadores, como também pode ser necessário mais de um controlador para reger um único atuador. Controladores são especificados para cada uso e são criados pelos programadores dos dispositivos e interface, o *middleware* não restringe quais controladores um dispositivo pode conter, essa informação é repassada para a interface e ela irá exibir os controladores existentes.

A Listagem 2 mostra os controladores cadastrados em um dispositivo de exemplo, essas informações são armazenadas pelo servidor e repassadas para os serviços que utilizam o *middleware*, como uma interface.

```
{
  "controllers": [
    {
      "key": "port1",
      "type": "onoff",
      "tag": "liga e desliga",
      "description": "liga e desliga a luz"
    },
    {
      "key": "port2",
      "type": "rgbx",
      "tag": "cor",
      "description": "muda a cor da luz"
    },
    {
      "key": "port3",
      "type": "toggle",
      "tag": "motor",
      "description": "Liga um motor"
    },
    {
      "key": "port4",
      "type": "button",
      "tag": "cor aleatoria"
    }
  ]
}
```

Listing 2: JSON de exemplo de controladores de um dispositivo

3.2.2 Sensor

Um sensor é formado por atributos, descritos na Listagem 3 ,como o tipo (*type*), que pode ser *number*, *boolean*, *point*, dentre outros, que define o tipo de dado enviado na comunicação; um sensor opera em estado de envio síncrono e assíncrono simultaneamente, ou seja, por intervalo de tempo (*timeout*) ou por extrapolação do valor de sua variação da medição do ultimo envio e atual medição excedendo o *deadBand*; caso esteja em regimento síncrono, o sensor forma pacotes de dados com intervalo de tempo com uma resolução (*resolution*); alguns tipos necessitam de informações adicionais como o tipo *number*, que em alguns casos possui um funcionamento acumulado (*accumulate*) informando se os dados são acumulados com o tempo ou se são valores instantâneos, possibilitando o servidor acumular e o dispositivo enviar somente o diferencial; e a unidade de medida (*unit*) na qual o sensor funciona e é visualizada pelo usuário.

```
{
  "sensors": [
    {
      "key": "senDev1",
      "deadBand": 300,
      "timeout": 60000,
      "resolution": 600,
      "type": "number",
      "accumulate": true,
      "tag": "hidrômetro geral",
      "unit": "litros"
    },
    {
      "key": "sensDev2",
      "timeout": 100000,
      "resolution": 10000,
      "type": "boolean",
      "tag": "porta entrada",
      "unit": "presença",
      "description": "porta da entrada"
    },
    {
      "key": "sensDev3",
      "timeout": 100000,
      "resolution": 5000,
      "type": "point",
      "tag": "gps carro",
      "unit": "coordenadas",
      "description": "gps do carro"
    }
  ]
}
```

Listing 3: JSON de exemplo de controladores de um dispositivo

3.2.3 Dispositivo

Um dispositivo é definido como um conjunto de controladores e sensores além de informações que ditarão sua comunicação com servidores. Algumas propriedades internas aos sensores e atuadores permitem a troca de dados com serviços que utilizam a padronização. Alguns desses são: a chave (*key*) é uma identificação única para cada controlador ou sensor em um dispositivo que juntamente com a identificação do serial, permite que

as trocas de informações entre serviços sejam otimizadas.

3.2.4 Produto

Para permitir que um produto seja usado no controle ou monitoramento de vários equipamentos, um produto é definido como um conjunto de dispositivos, barateando, por exemplo, aparatos para medição de água de uma rede de hidrômetros em que um mesmo produto (*hardware*) composto por vários dispositivos pode ser ligado a cada um dos hidrômetros, não sendo necessário a aquisição de um produto dedicado a cada hidrômetro.

3.3 Implementação

Primeiramente foi idealizada uma aplicação para automação composta por sensores e atuadores que comunicam-se com um servidor ou um minicomputador.

Foram escolhidos microcontroladores dotados de conectividade *Wireless* IEEE802.11 o qual possui capacidade de conexão nos IEEE802.11/b/g/n para compor os dispositivos com sensores e atuadores.

Posteriormente foram escolhidas as linguagens de programação e *frameworks* para o desenvolvimento de cada parte que comporia o sistema. Em seguida foi desenvolvido um sistema web conforme descrito na seção anterior para possibilitar a usuários tanto o controle quando o monitoramento dos dispositivos incluídos no sistema.

Após a escolha dos componentes e da infraestrutura montada foram idealizados alguns padrões de como as mensagens devem ser organizadas e como elas são trocadas para não haver desperdício no uso de rede. A Listagem 4 demonstra como a aplicação envia dados para o controlador específico de um dispositivo, é enviado a identificação serial do dispositivo (serial) e o nome do campo com o valor (59) que será enviado recebe o nome da chave do controlador que será acionado ("a1").

```
{
  "serial": "numero_de_serie",
  "a1": 59
}
```

Listing 4: JSON de envio de valores para um controlador de um dispositivo

A Listagem 5 representa o formato que um sensor de um dispositivo qualquer deve seguir para reportar o valor captado para uma aplicação servidora. Assim como na atuação de um controlador, é necessário a identificação serial do dispositivo (*serial*), da chave do sensor ("a1") que realizou a medição, do valor (50) que foi captado e da data (*dateTime*) que tal medição foi realizada.

```
{
  "serial": "numero_de_serie",
  "key": "a1",
  "value": 50,
  "dateTime": "2018-03-12 11:03:15"
}
```

Listing 5: JSON de envio de valores medidos pelo sensor de um dispositivo

4 Resultados e discussões

Os testes demonstraram que a proposta possibilita a criação de uma plataforma genérica para dispositivos inteligentes baseados em Internet das coisas com tipagem fraca, possibilitando um desenvolvimento rápido de sistemas embarcados sem a necessidade de adequação específica da arquitetura ou do servidor para que um novo dispositivo seja compatível. As escolhas específicas de desenvolvedores para as camadas abaixo de aplicação ou a sua topologia de rede não interferem no serviço disponível, tornando sólida a proposta de dividir as responsabilidades para as arquiteturas dos dispositivos e aplicações.

Figura 2: Interface de usuário mostrando a renderização dos controladores de um dispositivo

A varificação foi feita com alguns dispositivos compostos de atuadores e/ou sensores, tais como: hidrômetro que reporta o consumo de água, tomada que contabiliza corrente elétrica, lâmpadas inteligentes para acionamento remoto, automóvel que envia dados do seu computador de bordo e, sensor de batimento cardíaco. Esses dispositivos

inteligentes, capazes de enviar seus dados e receber comandos de uma aplicação em um servidor disponível na nuvem, usaram a padronização proposta e puderam se comunicar de acordo com escolhas feitas pelos usuários finais. A figura 2 demonstra a renderização de um dispositivos com alguns controladores.

5 Conclusões

Com o teste implementado em uma arquitetura de Internet das coisas foi possível concluir a expressividade da proposta em um meio de cooperação entre dispositivos inteligentes. A implementação foi feita para centralizar as tomadas de decisão, cadastro, controle e analise dos equipamentos conectados, melhorando assim a confiabilidade e experiência para indústrias, empresas ou casas que utilizarão desse serviço.

A partir da descrição dos dispositivos, estudantes ou empresas podem desenvolver produtos capazes de se conectar a uma rede de máquinas e aplicações, o que facilita a disseminação da Internet das coisas. Não há a necessidade de implementar novamente características de um serviço ou padrão de comunicação já feito, tornando mais simples o processo de elaboração de aparatos inteligentes. Desenvolvedores de sistemas embarcados podem focar seus esforços para conquistar uma necessidade de um nicho de mercado sem mais contratar programadores de alto nível de abstração para satisfazer suas necessidades na camada de aplicação.

Para futuras implementações e constatações serão desenvolvidos dispositivos com características ainda não testadas usando protocolos de transporte e aplicação diversos. Testes usando computação na borda (*edge computing*) e computação nebulosa (*fog computing*) serão feitos para demonstrar que escolha da topologia de rede ou onde está alocado o poder de processamento dos dados irá depender da aplicação e sua necessidade de computação de dados.

Referências

Acciari, G., Caruso, M., Miceli, R., Riggi, L., Romano, P., Schettino, G. and Viola, F. (2018). Piezoelectric rainfall energy harvester performance by an advanced arduino-based measuring system, *IEEE Transactions on Industry Applications* **54**(1): 458–468.

Atzori, L., Iera, A. and Morabito, G. (2010). The internet of things: A survey, *Computer networks* **54**(15): 2787–2805.

Augustin, A., Yi, J., Clausen, T. and Townsley, W. M. (2016). A study of lora: Long range & low power networks for the internet of things, *Sensors* **16**(9): 1466.

Centenaro, M., Vangelista, L., Zanella, A. and Zorzi, M. (2016). Long-range communications in unlicensed bands: The rising stars in the iot and smart city scenarios, *IEEE Wireless Communications* **23**(5): 60–67.

Christensen, K., Reviriego, P., Nordman, B., Bennett, M., Mostowfi, M. and Maestro, J. A. (2010). Ieee 802.3az: the road to energy efficient ethernet, *IEEE Communications Magazine* **48**(11): 50–56.

Crockford, D. (2006). The application/json media type for javascript object notation (json).

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T. (1999). Hypertext transfer protocol–http/1.1, *Technical report*.

Kurose, J. F. (2013). *Redes de computadores e a Internet: uma abordagem top-down*, reprint edn, Pearson Education do Brasil Ltda.

Macêdo, W. L. d. A. M. et al. (2016). Gothings: uma arquitetura de gateway de camada de aplicação para a internet das coisas.

McCormick, D. K. (2017). Ieee technology report on wake-up radio: an application, market, and technology impact analysis of low-power/low-latency 802.11 wireless lan interfaces.

Petrova, M., Wu, L., Mahonen, P. and Riihijarvi, J. (2007). Interference measurements on performance degradation between colocated ieee 802.11 g/n and ieee 802.15. 4 networks, *Networking, 2007. ICN'07. Sixth International Conference on*, IEEE, pp. 93–93.

Siekkinen, M., Hiienkari, M., Nurminen, J. K. and Nieminen, J. (2012). How low energy is bluetooth low energy? comparative measurements with zigbee/802.15.4, *2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 232–237.

Sigfox S.A. (2018). Sigfox technology overview. acessado em 15 de abril de 2018.
URL: *https://www.sigfox.com/en/sigfox-iot-technology-overview*

Standard, O. (2014). Mqtt version 3.1. 1, *URL http://docs. oasis-open. org/mqtt/mqtt/v3* **1**.

Tozlu, S., Senel, M., Mao, W. and Keshavarzian, A. (2012). Wi-fi enabled sensors for internet of things: A practical approach, *IEEE Communications Magazine* **50**(6).

Yokotani, T. and Sasaki, Y. (2016). Comparison with http and mqtt on required network

resources for iot, *Control, Electronics, Renewable Energy and Communications (ICCEREC), 2016 International Conference on*, IEEE, pp. 1–6.