SISTEMA ROBÓTICO BASEADO NUMA ARQUITETURA DE IOT, UTILIZANDO ROS, SOA E IBM WATSON.[1]

Henrique Conti Garcez[2]

Élder Bernardi[3]

João Mario Brezolin[4]

RESUMO

Uma tendência que está acontecendo no estabelecimento da era digital é o de busca, captação e tradução de dados presentes no ambiente externo. Conceituado como Internet das Coisas, ele possibilita e favorece o reuso de suas aplicações e interoperabilidade com outras tecnologias. Nesse mérito, uma das possibilidades que apresenta maior nexo e compatibilidade é a integração dos conceitos de IoT com a robótica já que ambas estão presentes em diversos escopos, porém, em muitos casos, ainda permanecem segmentadas. Objetivando estabelecer a interoperabilidade entre componentes e estruturas das tecnologias, o trabalho que segue propõe um modelo de sistema de robótica baseado em uma arquitetura de IoT. Onde cada componente do sistema possa de alguma forma se encaixar e suprir as funcionalidades da arquitetura apresentada. No que se refere à construção do sistema, ferramentas e metodologias como o ROS, SOA e o Watson da IBM foram utilizadas como alicerces no desenvolvimento do projeto. No propósito de chegar a um sistema final, não só coerente quanto à arquitetura de IoT apresentada, mas também apto à aplicação de tecnologias emergentes. Um protótipo foi construído para validar a proposta do trabalho, sendo avaliado pela forma que cada componente da sua estrutura se encaixou na arquitetura de IoT referência, assim como pela eficiência e capacidade de comunicação dos componentes. Considerando as métricas de avaliação, os resultados obtidos pelo trabalho foram satisfatórios. Conclui-se que a união das tecnologias pode ser proveitosa e capaz de proporcionar a criação de sistemas mais robustos e menos segmentados.

Palavras-chave: Internet das coisas, robótica, integração, arquitetura, tecnologias emergentes.

---

[1] Trabalho de Conclusão de Curso (TCC) apresentado ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-rio-grandense, Câmpus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet, na cidade de Passo Fundo, em 2018

[2] Estudante de Sistemas para Internet no IFSul, natural de Passo Fundo, email: <henrique.c.96@hotmail.com>

[3] Orientador, professor do IFSUL, email: <elder.bernardi@passofundo.ifsul.edu.br>

[4] Co-orientador, professor do IFSUL, email: <joao.brezolin@passofundo.ifsul.edu.br>.

1 INTRODUÇÃO

Desde a idealização da Internet das Coisas, surgiu a noção de que mais cedo ou mais tarde a maioria dos elementos físicos (palpáveis), que fazem parte do nosso cotidiano, estarão de alguma forma conectados à rede (Internet). Gubbi afirma “No paradigma da Internet das coisas, muitos objetos que nos cercam estarão na rede de um jeito ou de outro.”(GUBBI et. al, 2013).

Nesse contexto, é notável que alguns objetos parecem ter mais afinidade com a integrabilidade proposta pela IoT(Internet das coisas). Como por exemplo, dispositivos robóticos. Dentre eles, um dos que mais vem recebendo destaque e incentivo nos últimos anos é o carro autônomo. A CBInsights contabilizou em uma de suas pesquisas “[...]nós identificamos 46 companhias desenvolvendo carros autônomos. As empresas variam de diversas áreas, desde automotivas até empresas de telecomunicação[...] ”(CBINSIGHTS, 2018). Números que ilustram bem o desafio e as oportunidades na confecção de ferramentas e estruturas focadas na melhoria da interoperabilidade entre as tecnologias.

Considerando esse cenário, associado à motivação de trabalhar com assuntos relacionados às áreas da robótica e IoT, propõe-se a construção de um modelo de arquitetura para robôs, baseado na arquitetura de IoT. Tentando assim, explanar uma forma de integração de dispositivos genéricos a robóticos, similarmente ao caso do carro autônomo.

Referente à proposta de trabalho, tal modelo tem por objetivos apresentar uma arquitetura de IoT referência e as principais tecnologias utilizadas para a construção do trabalho: ROS, IBM WATSON E SOA, desenvolver um sistema robótico embasado na modelagem e na forma de comunicação dos componentes da arquitetura, elencando métodos e ferramentas capazes de atuar em funções semelhantes aos componentes da arquitetura apresentada, assim como demonstrar possíveis cenários de aplicação do sistema.

A avaliação do sistema será dada através da implementação de um protótipo sobre um cenário proposto, para verificar a atualização dos componentes da arquitetura apresentada para o escopo da robótica e também, avaliar a troca de informações entre as partes do sistema.

O restante do artigo está assim organizado: na Seção 2 a arquitetura de IoT tomada como base será apresentada; na Seção 3 será elucidado o modelo proposto, traçando um paralelo entre a arquitetura e o sistema; a Seção 4 comentará sobre alguns trabalhos relacionados; na Seção 5 o sistema robótico elaborado será descrito e avaliado e, por fim, na Seção 6 as considerações quanto ao resultado do projeto serão expostas.

2 ARQUITETURA DE IOT E FERRAMENTAS

Discorrendo sobre a proposta do modelo que será apresentado, infere-se a exposição dos elementos estimados como alicerces para construção do projeto. Dentre eles, as ferramentas, serviços e, sobretudo, a arquitetura de IoT tomada como referência para a construção do sistema.

2.1 IOT

Termo criado por Kevin Ashton, a internet das coisas pode ser descrita como a análise de dados coletados por um *hardware*(equipamento) em comunicação direta com o *software* (aplicação)(LUZ; HUMENHUK, 2015). Assunto que foi explanado pelos autores no curso Internet das Coisas – Fundamentos de IoT.

O processo se inicia da necessidade de coleta de informação em algum ambiente, em alguma “coisa”. Como por exemplo, a necessidade da medição de temperatura, em tempo real, num ambiente fechado para o cultivo de certa hortaliça. Com a definição de escopo e dado a ser coletado também é possível elencar o equipamento necessário para a realização de tal tarefa, como sensores e dispositivos capazes de coletarem os dados presentes e disponíveis no ambiente externo à sistemas digitais. e transformá-los para uma forma mensurável.

A partir da obtenção dos dados feita por um dispositivo, infere-se sua aplicação na estrutura de IoT que está contido (arquitetura), no intuito de alcançar melhorias e adaptação sobre os dados obtidos.

2.2 ARQUITETURA DE IOT

Na busca por uma arquitetura de IoT compatível com os objetivos a serem alcançados, algumas diretrizes foram elencadas quanto à forma que cada componente da estrutura poderia ser modificado ou substituído para melhor atender o novo modelo. Entre elas, uma camada de tradução e representação de dados, uma camada de transporte e abstração de dados, e, por fim, uma camada para processamento dos dados coletados.

Considerando esse cenário e a interoperabilidade proposta pelo modelo, verificou-se também a necessidade de opção por uma arquitetura de IoT capaz de ser dividida como uma coleção de serviços.

Com base nos requisitos citados anteriormente, a arquitetura apresentada por Kasturi se mostrou proveitosa,

- *Device layer*(Camada do dispositivo)
- *Access management layer* (Camada de gerenciamento)
- *Transport layer* (Camada de transporte)
- *Event processing layer* (Camada de controle de ação)
- *Application layer* (Camada da aplicação) (KASTURI, 2016)

Refletindo a propósito da função de cada um dos componentes e da demonstração da estrutura em um cenário padrão IoT. Infere-se a decomposição dos elementos da arquitetura.

Começando pela conexão(laço) entre o modelo computacional e objetos dispostos no ambiente externo, a camada do dispositivo tem o papel de coletar os dados brutos oriundos de fora, exemplificada por sensores e dispositivos capazes de coletar dados do ambiente.

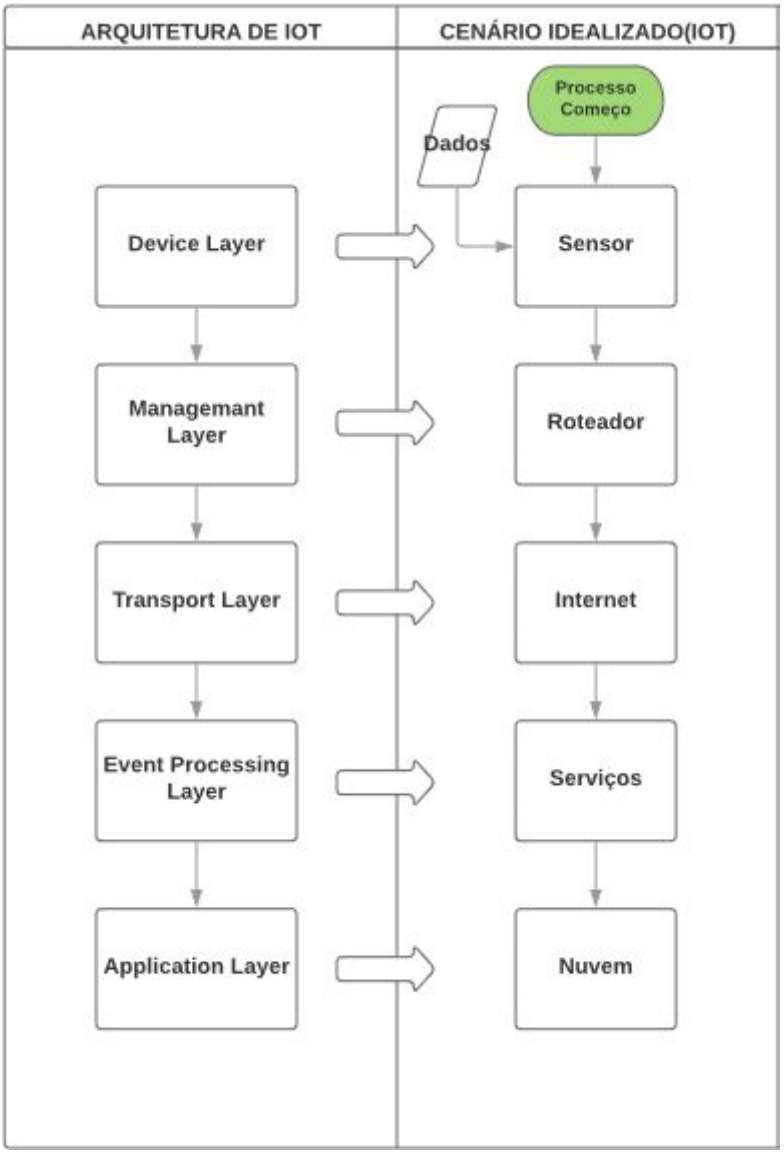
Quanto à camada de gerenciamento, denota-se sua característica pela capacidade de controle e tradução dos dados oriundos do sensor. Atuando como *gateway* para a passagem e representação dos dados para o entendimento da rede. No cenário de IoT, um bom exemplo de dispositivo capaz de exercer essa função é o roteador.

Com o pressuposto de que os dados já estão compreensíveis a nível computacional, a terceira camada da arquitetura se encarrega de enviar ou dissipar os dados para diversos serviços, como no caso da Internet. Complementarmente à essa etapa, a camada de evento (ação) se caracteriza por operar como segregadora da fonte geradora de dados.

Finalmente, a camada da aplicação tem o papel de processar os dados coletados. Trabalhando de forma eficiente sobre os dados recebidos do sensor, no intuito de aproveitá-los da melhor maneira possível. A nuvem é um ótimo exemplo de representante da camada em um cenário IoT.

A Figura 1 ilustra um possível cenário de IoT baseado na estrutura da arquitetura de IoT apresentada.

Figura 1 - Cenário de IoT idealizado em conformidade com a estrutura da arquitetura



Fonte: Do Autor(2018).

Com a visão sucinta da arquitetura de IoT usada como alicerce para a confecção do projeto, infere-se a apresentação das principais tecnologias(ferramentas) que serão utilizadas na construção do modelo.

2.3 ROS (ROBOT OPERATIONAL SYSTEM)

Desenvolvido por pesquisadores da *Willow Garage*[5]*. O* ROS é um *framework*[6] para construção de aplicações para robôs munido de uma robusta biblioteca de

[5] Incubadora de desenvolvimento de hardware e software para aplicações robóticas.(WILLOW GARAGE).
[6] Estrutura base para a construção de alguma coisa. (CAMBRIDGE DICTIONARY).

ferramentas auxiliares. Os próprios criadores da plataforma elucidam “ROS é um conjunto de bibliotecas e ferramentas que ajudam na construção de aplicações para robôs. De drivers até algoritmos de última geração e com poderosas ferramentas de desenvolvimento”(ROS, 2017).

Sua estrutura de aplicação se baseia na troca de mensagens entre os “pedaços” do *framework* através de tópicos. De acordo com Krahel

> O ROS é baseado em nódulos, mensagens, tópicos e serviços. No ROS, nódulos são os módulos de software ou processos no código de controle. Eles se comunicam uns aos outros através da passagem de mensagens. Um tipo especial de mensagem, chamado serviço, consiste em um par de mensagens, uma para requisição e a outra para a resposta. Os nódulos podem publicar e assinar um único (ou vários) tópicos. (KRAHEL et. al.)”tradução nossa”.

No que se refere ao empenho das ferramentas auxiliares presentes no escopo do ROS, muitas delas atuam na abertura e facilitação da comunicação do ROS com dispositivos, sistemas e redes externas ao seu controle. Considerando essa premissa, infere-se a apresentação das principais ferramentas e bibliotecas utilizadas na modelagem do ROS concernente ao sistema proposto.

2.3.1 ROSJAVA

Biblioteca que possibilita o emprego de Java puro, e consequentemente, a integração de dispositivos Android ao escopo de nódulos do ROS.(ROSJAVA, Acesso em: 10 mai. 2018). Permitindo até mesmo a passagem de dados coletados por sensores do celular para o *framework*, como Aroca verifica “[...] dados coletados por sensores de dispositivos Android, como dispositivos internos, bússola e câmera, podem ser publicados em tópicos do ROS.”(AROCA et. al, 2012).

2.3.2 ROSBRIDGE

Constitui-se de uma interface de comunicação para programas externos ao ROS(ROSBRIDGE, 2018). Atuando como *middleware* entre nódulos do ROS e outras tecnologias externas ao seu escopo. Cabe ressaltar que as mensagens trocadas entre as partes são objetos formatados em JSON[7]. Blaha afirma “Rosbridge permite clientes publicarem e assinarem mensagens de tópicos e invocar serviços no ambiente de execução. Rosbridge transporta mensagens formatadas em JSON por Soquetes TCP e web-sockets.”(BLAHA et. al, 2013).

---

[7] JavaScript Object Notation, formatação para troca de dados baseada na linguagem JavaScript.(JSON)

2.4 SOA

No intuito de propor um modelo coerente, maleável e facilitador à integração com tecnologias emergentes, optou-se por projetá-lo não somente embasado na arquitetura de IoT apresentada, mas também englobando algumas metodologias de construção de *software* para melhor organização do projeto. Nesse sentido, o conceito de *Service Oriented Architecture*(SOA, traduzido como Arquitetura Orientada a Serviços) e o método *publisher/subscriber* foram as principais escolhas. Sendo que o último é um padrão cada vez mais utilizado em sistemas IoT, averiguado por Ogliari

> "Recentemente, um padrão vem ganhando destaque no mundo do desenvolvimento, principalmente quando falamos de mobile e Internet of Things. O padrão Publisher/Subscriber está cada vez mais presente em APIs, frameworks e bibliotecas de código."(OGLIARI, 2017).

Contextualizando os critérios adotados, a especificação do método *publisher/subscriber* é dada similarmente à estruturação da comunicação entre processos executada pelo ROS, já a sistemática do SOA é definida por Papazoglou como a implementação de serviços para o agrupamento de sistemas distribuídos, sendo que a arquitetura provém muita flexibilidade quanto à interoperabilidade de seus componentes(Papazoglou, 2013).

2.5 IBM WATSON

Plataforma criada pela IBM[8], que facilita a construção de modelos e classificadores baseados em *machine learning*[9] - processados na nuvem.(IBM WATSON). Repleta de possibilidades a plataforma permite a criação de modelos customizados para diferentes algoritmos de aprendizagem. O envio de dados para tais modelos se dá através de serviços disponibilizados pela ferramenta.

O principal foco do trabalho é para a API de reconhecimento de imagem que utilizada *deep learning*[10] para análise e classificação de imagens(IBM WATSON VISUAL RECOGNITION SERVICE, 2018).

---

[8] International Business Machines Corporation, é uma multinacional americana focada em tecnologias da informação.

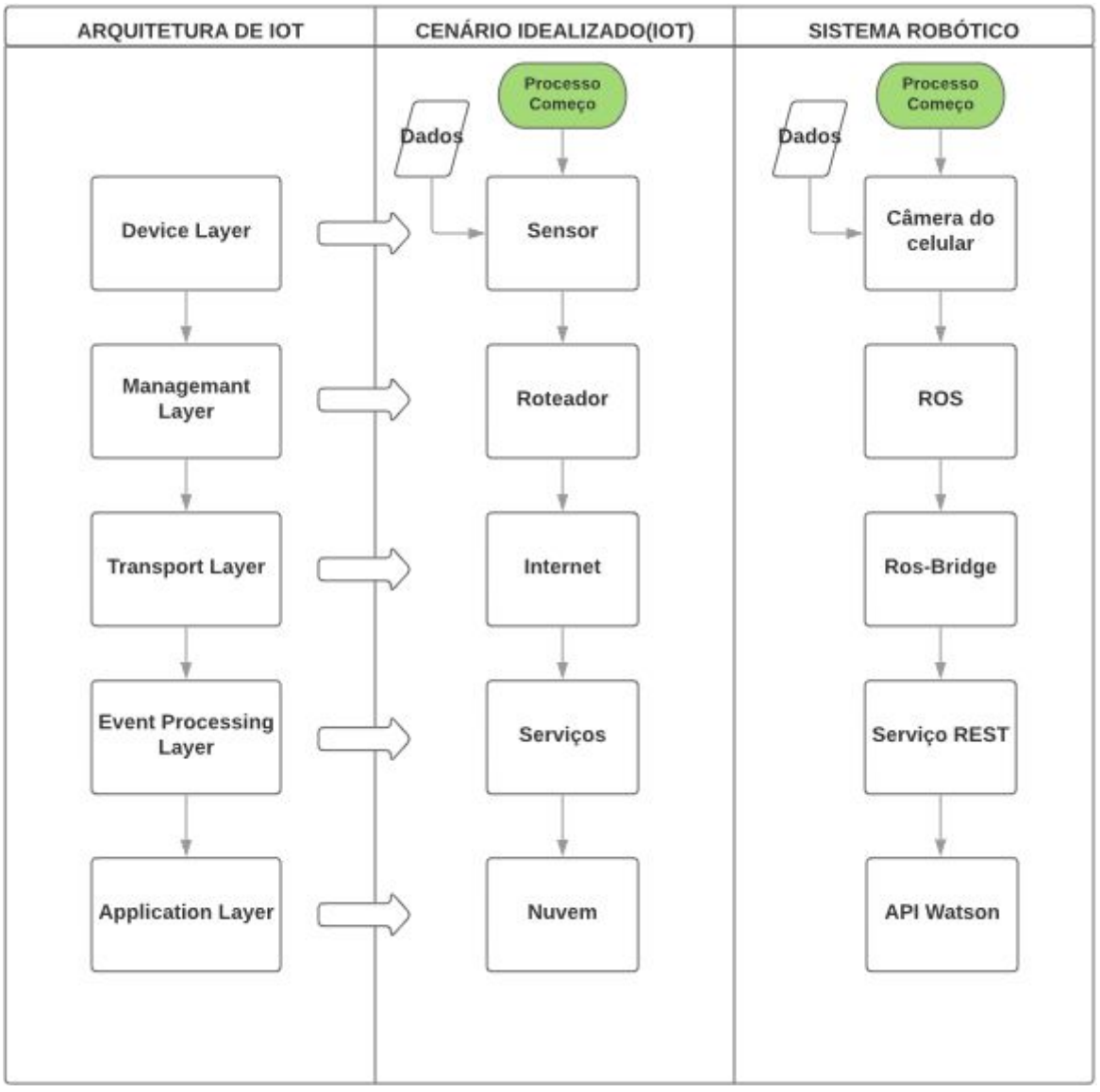
[9] Área da inteligência artificial que elucida o aprendizado computacional através de algoritmos que possibilitam a adaptação e evolução do código perante treinamento.

[10] Subseção do aprendizado de máquina: é o campo mais avançado na área da inteligência artificial e o que mais aproxima a máquina de aprender e pensar como um ser humano.(DE JESUS, 2017).

3 MODELO PROPOSTO

No que se refere ao embasamento do modelo à arquitetura de IoT apresentada, resta elucidar a atualização dos componentes e sua função em conformidade às características de um sistema robótico. O paralelo traçado entre o cenário de IoT idealizado e o novo modelo está ilustrado na Figura 2.

Figura 2 - Paralelo entre cenário idealizado e sistema proposto



Fonte: Do Autor(2018).

Sobre a nova estrutura criada, quanto às alterações de componentes do cenário de IoT para sistema robótico, as camadas da arquitetura ficaram estruturadas da seguinte forma: A *device layer* é responsável pela extração de dados do meio externo, foi representada por uma câmera celular ou coletor de

imagem; A *access management layer,* etapa de passagem, controle e representação dos dados, foi representada pelo ROS - sistema que também representará a abstração do robô; A *transport layer* e a *event processing layer,* responsáveis pela comunicação e abstração dos dados provenientes do dispositivo, foram representadas por dois micro-serviços distintos - o primeiro para evidenciar a comunicação do sistema com a camada de gerenciamento e o segundo para exemplificar o isolamento da rede à camada da aplicação; A *application layer,* encarregada de executar o processamento dos dados coletados de maneira inteligente, foi representada pela API do Watson - que classifica as imagens coletadas em conjunto com a nuvem.

Ainda é possível descrever um cenário real de aplicação do modelo, conceituando: o sistema poderia ser adequado por um "robô vigia" (robô com câmera acoplada), que cuidasse de uma via de pedestres. Ao perceber uma situação de risco através da classificação inteligente das imagens coletadas pelo sensor, o ROS teria o controle sobre o robô para contornar a situação de maneira adequada.

Na próxima Seção do trabalho será feita a análise sobre a implementação da estrutura confeccionada, argumentando sobre um cenário ideal do funcionamento do protótipo do sistema.

4 PROTÓTIPO DESENVOLVIDO

Ponderando sobre o comportamento esperado do sistema em um caso de teste, o protótipo funciona da seguinte forma: ao capturar uma imagem, a câmera de celular dispararia o gatilho de ativação do sistema; o fluxo de transmissão dos dados publicaria a informação (imagem) em um processo subsidiado pelo ROS; concebida a noção de controle e representatividade dos dados ao ROS, ele daria sequência ao fluxo do sistema; a camada de transporte (RosBridge) permitiria a dissipação dos dados para serviços e programas externos ao escopo do ROS - efetivando a requisição para o serviço REST[11]; a camada de ação isolaria a fonte do dado oriundo da camada do transporte para o restante do sistema e utiliza métodos de chamada para controle da camada de aplicação (API do Watson); A API faria então a classificação (processamento) da imagem coletada e a requisição retornaria o resultado para a camada de transporte; o fluxo do protótipo chega ao fim quando a camada de transporte informa ao ROS o resultado processado pela camada de aplicação e o ROS toma uma ação de acordo com o resultado obtido.

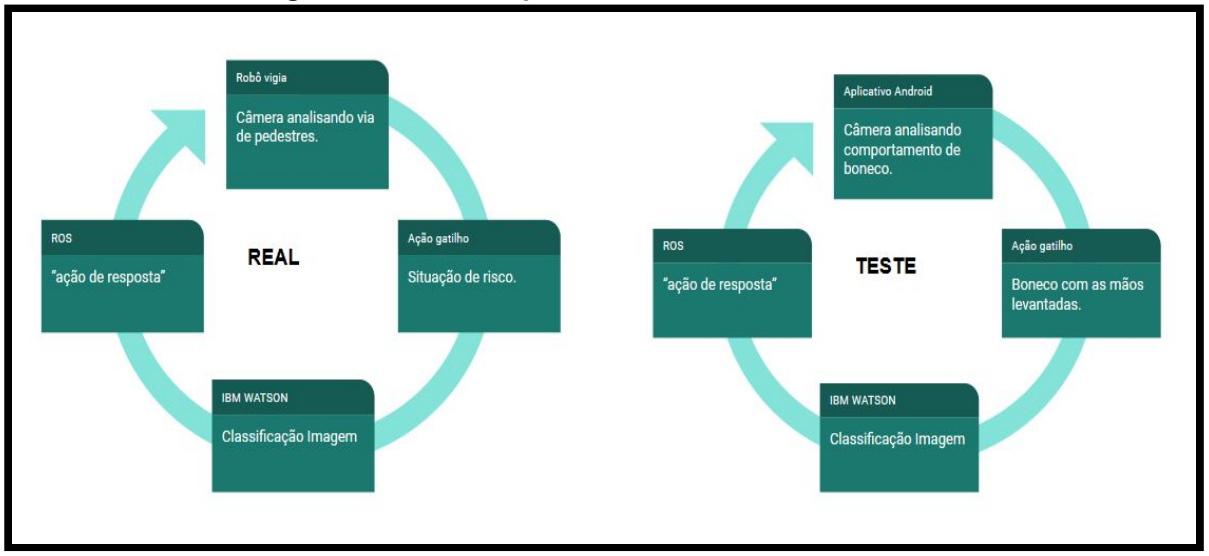
Buscando verificar a aplicabilidade do protótipo em uma situação representativa do cenário real ilustrado anteriormente, foi articulado o seguinte experimento: A câmera do celular é direcionada para um boneco estático em uma

[11] Metodologia de criação de web services baseado nos métodos HTTP (GET, POST, PUT e DELETE)(REST API TUTORIAL, 2018).

superfície, representando a iteração do sistema com uma pessoa; O sistema então responde aos movimentos e formação de membros feitos pelo boneco; Um exemplo de conjuntura válida e treinada pela aplicação é o de membros levantados pelo boneco, indicando uma situação de perigo em que o sistema precisa intervir para garantir a segurança dos elementos presentes na ocorrência; Quando o boneca age de tal forma o sistema verifica a mudança de comportamento através da classificação da nova imagem e o ROS responde ao publicar uma mensagem de alerta em um tópico de controle do robô.

A Figura 3 ilustra o fluxo comparativo entre os dois cenários descritos.

Figura 3 - Fluxo comparativo entre o cenário real e o de teste.



Fonte: Do Autor(2018).

Com a visão do sistema robótico e a noção do comportamento ideal do sistema, a próxima Seção trata de descrever e avaliar a execução do sistema proposto.

5 AVALIAÇÃO DO SISTEMA ROBÓTICO

Quanto a descrição do sistema: Assim como em qualquer escopo de implementação do conceito de IoT, o processo que incita a ativação do sistema é a conexão da rede interna com o meio externo. Nesse modelo, uma câmera de celular, controlada por um aplicativo, atua como elo entre os dados do ambiente externo e o sistema robótico. O processo de captação de dados é caracterizado pela produção fotográfica sobre a visão da lente do sensor.

Sobre a função do aplicativo, denota-se o gerenciamento do período em que cada foto é obtida e da passagem dos dados brutos (imagem em matriz de bytes) para o meio robótico. Tal comunicação é feita através da publicação dos dados em um tópico do ROS. Vale ressaltar que a confecção do aplicativo citado não foi de

autoria própria, mas sim baseado sobre um aplicativo molde da biblioteca Rosjava presente na documentação do *framework*.

Responsável pela tradução e representação dos dados para o senso robótico, o ROS atua como tradutor dos dados para camadas superiores, assim como abstrai a representação do próprio robô. Contextualizando, no sistema proposto, além de ter o papel de *gateway* entre os dados coletados e a rede interna, o ROS também terá a função de executar ações baseadas nos resultados obtidos pela etapa de processamento de dados (camada de desenvolvimento).

Quanto ao tratamento dos dados oriundos do sensor para a camada de transporte: enquanto os dados são publicados em um tópico '/camera' pelo aplicativo, outro processo, *subscriber* desse tópico, faz a conversão da matriz de bytes para um formato legível e transferível no âmbito da rede, como por exemplo: .jpeg. Tal processo(nódulo) é concretizado com o auxílio da biblioteca OpenCV[12].

A Figura 4 demonstra um trecho de código do nódulo de formatação da imagem, escrito em C++.

Figura 4 - Trecho do código do nódulo de conversão de imagem do ROS

```
public:
  ImageConverter()
    : it_(nh_)
  {
    // Escutando ao tópico /camera
    image_sub_ = it_.subscribe("/camera/image/", 1,
      &ImageConverter::imageCb, this, image_transport::TransportHints("compressed"));
    // publicando imagem formatada no tópico /image_converter
    image_pub_ = it_.advertise("/image_converter/output_video", 1);

  }

  ~ImageConverter()
  {
    std::cout << "Gravou ERRO";
    cv::destroyWindow(OPENCV_WINDOW);
  }

  void imageCb(const sensor_msgs::ImageConstPtr& msg)
  {
    cv_bridge::CvImagePtr cv_ptr;
    std::cout << "Gravou";

    cv_ptr = cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::BGR8); // conversor de imagem

  }
};
```

Fonte: Do Autor(2018).

A conversão da imagem possibilita a continuidade no fluxo de funcionamento do sistema. Com o dado maleável, o nódulo envia o objeto formatado para a camada de transporte. Comunicação que é feita através do tópico 'image_converter', também descrito na Figura 4.

[12] Open Source Computer Vision Library, é uma biblioteca de programas de visão computacional e aprendizado de máquina em código aberto(OPENCV, 2018).

No mérito da camada de transporte, ela é constituída por um serviço web-socket que é executado sobre a rede interna em uma porta qualquer, atuando como middleware de mensagens(MOM). Codificado para responder a chamadas no tópico ‘image_converter’ - o servidor tem uma função de callback(subscriber) apontada para o tópico do ROS. No que tange à construção desse serviço, foi concretizada graças ao molde disponibilizado pela biblioteca ros-bridge, também disponível na documentação do ROS.

Quanto à função de retorno do servidor: recebendo o objeto formatado(imagem) no formato JSON como parâmetro, faz a dissipação dos dados para a camada de abstração - evidenciada pela requisição HTTP do tipo POST ao serviço REST. A Figura 5 apresenta o comportamento do servidor web-socket escrito em Node.JS[13].

Figura 5 - Demonstração ilustrativa do funcionamento do web-socket

```
// Estabelece conexão com o servidor Web-Socket do RosBridge.
ros.connect('ws://192.168.0.17:9090');

// Cria tópico /image_converter e /response
var fObj = new ROSLIB.Topic({
  ros : ros,
  name : '/image_converter',
});

var rObj = new ROSLIB.Topic({
  ros : ros,
  name : '/response',
});

//Escuta a passagem de dados pelo tópico /image_converter
fObj.subscribe(function(message) {
  request.post(message, function (success) { // simulação request para serviço REST
    // publica resposta no tópico /response
    rObj.publish(success.message);

  });

});
```

Fonte: Do Autor(2018).

No que se refere à justificativa da requisição para o serviço REST, denota-se a caracterização dele como representante da camada de ação. Vale ressaltar que a partir dessa camada, por se tratar da etapa de abstração da fonte de dados, o funcionamento do sistema não depende mais do meio robótico. Em outras palavras, qualquer *software (*programa de computador) também poderia usufruir do

[13] Ambiente de execução JavaScript, assíncrono e baseado em eventos(NODEJS, 2018).

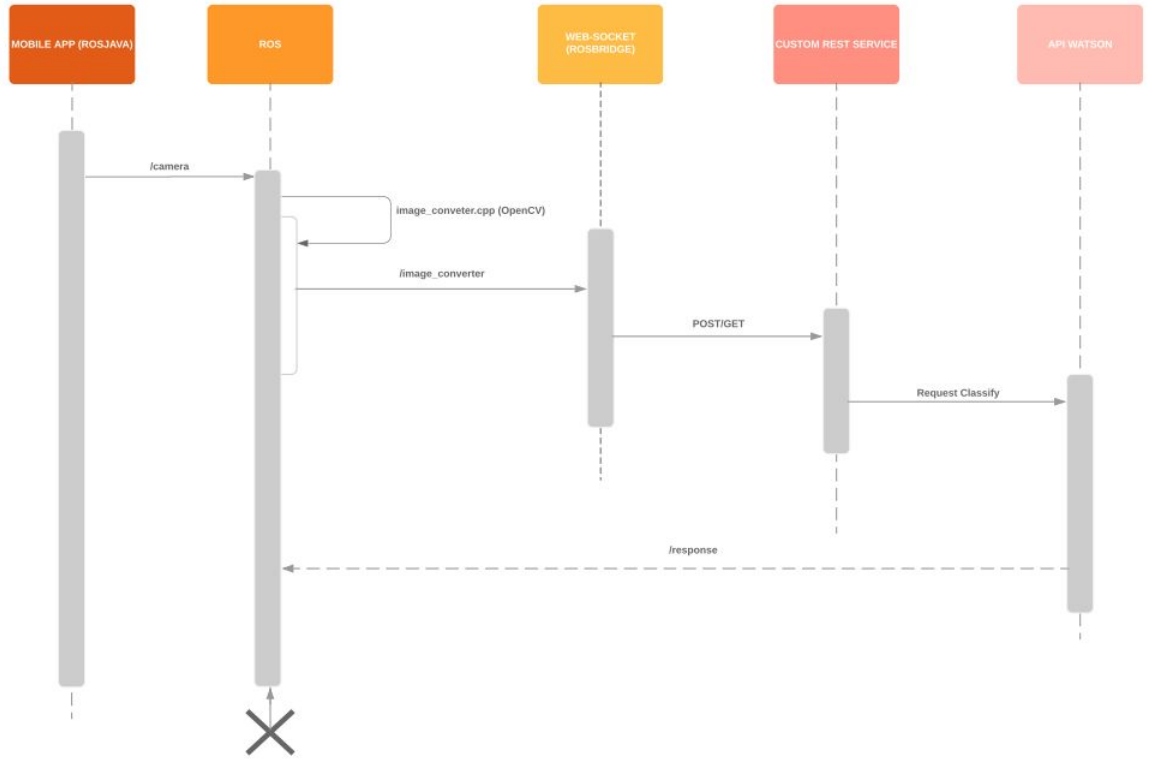
web-service e da camada de processamento, sem perda ou alteração no condicionamento dos dados.

O serviço REST, agente da abstração do sistema, foi desenvolvido em Node.JS e através dos métodos HTTP, POST e GET, permite a comunicação com a camada de processamento. No caso do método POST, o web-service exige um objeto do tipo imagem no corpo da requisição, e se encarrega de enviá-lo para a camada de aplicação, onde está o classificador do Watson. Já no método GET, o web-service busca as informações genéricas referentes a camada de aplicação, como: status de operação e características da API.

Finalmente, no que tange ao funcionamento da camada de aplicação, o método POST faz uma requisição para um classificador de imagem customizado na API do Watson. Apresentada na Seção 2 do trabalho, a API possibilitou o treinamento de um classificador de imagem que conduz o processamento dos dados coletados.

O diagrama de sequência da Figura 6 ilustra o funcionamento do sistema.

Figura 6 – Diagrama de sequência do sistema proposto.



Fonte: Do Autor(2018).

No mérito da avaliação do sistema quanto ao comportamento esperado do protótipo desenvolvido, o funcionamento do sistema se mostrou coerente a estrutura de IoT apresentada. Analisando a execução do sistema: o aplicativo do ros-android publica no tópico ‘/camera’ do ROS as imagens capturadas pela câmera do celular; Um processo subsidiado do ROS verifica a passagem de dados pelo tópico e, depois

de formatá-los para o escopo da rede interna, faz a publicação destes para outro tópico ‘/image_converter’; O objeto(dado) é então externado do controle do ROS com o auxílio do web-socket Rosbridge; O servidor do socket executa uma chamada para o serviço REST passando o objeto formatado no corpo da requisição; O web-service, isolado quanto à proveniência do objeto, requisita o processamento dos dados pelo classificador customizado do Watson e retorna para o web-socket os resultados obtidos; Concluindo o experimento, o retorno da chamada é enviado para o tópico ‘/response’ do ROS através do web-socket;

6 CONSIDERAÇÕES FINAIS

O parecer final do trabalho elucidou resultados plausíveis sobre as métricas de avaliação adotadas. Entretanto, cabe ressaltar algumas dificuldades encontradas durante a preparação do protótipo, sendo que em maior parte os problemas surgiram na configuração de pacotes e bibliotecas externas a documentação do ROS.

Sobre a continuidade do trabalho, por se tratar de um sistema moldado em orientação a serviços, as partes do sistema poderiam ser isoladas e trabalhadas a parte, assim como o sistema inteiro também poderia ser continuado e melhorado. Almejando inclusive sua aplicação no cenário real exemplificado no decorrer do trabalho. Entre principais melhorias que podem ser aplicadas ao sistema, estão: a criação de banco de relatório no serviço REST; o desenvolvimento de interface web para interação do web-socket; a construção de um robô com câmera fotográfica integrada; Controle de ações do robô com o ROS.

ABSTRACT

One trend that is taking place in the establishment of the digital era is the search, capture and translation of data present in the external environment. Conceptualized as Internet of Things, it enables and favors the reuse of its applications and interoperability with other technologies. In this context, one of the possibilities of integration and reusability that emerges is the union of the concepts from IoT to robotics. In order to establish the interoperability between components and structures of the technologies, the following work proposes a robotic system based on an IoT architecture. Where each component of the system can somehow fit in and supply the features of the architecture presented. In terms of system building, tools and methodologies such as ROS, SOA and Watson from IBM were used as a foundation in the development of the project. In order to reach a final system, not only coherent with the IoT architecture presented, but also applicable with emerging technologies. A prototype was built to validate the work proposal, being evaluated by the way that each component of its structure fit the reference IoT architecture, as well by the efficiency and communication capability of the components. Considering the evaluation metrics, the results obtained by the work were satisfactory. It is concluded

that the union of the technologies can be useful and provide the creation of systems more robust and less segmented.

Keywords: Internet of Things, robotics, integration, architecture, emerging technologies.

REFERÊNCIAS

AROCA, Rafael V. et al. Towards Smarter Robots with Smartphones. 2012. Disponível em: <https://www.researchgate.net/profile/Luiz_Goncalves/publication/267843503_TOWARDS_SMARTER_ROBOTS_WITH_SMARTPHONES/links/5475ec8f0cf29afed612cde3.pdf>. Acesso em: 24 nov. 2018.

BLAHA, M. et al. Rosbridge web interfaces. 2013. Disponível em: <https://klein.felk.cvut.cz/w/_media/misc/projects/nifti/sw/web_interface-report.pdf>. Acesso em: 06 nov. 2018.

CAMBRIDGE DICTIONARY. Disponível em <https://dictionary.cambridge.org/>. Acesso em: 22 nov. 2018.

CBINSIGHTS. 46 Corporations Working On Autonomous Vehicles. 2018. Disponível em: <https://www.cbinsights.com/research/autonomous-driverless-vehicles-corporations-list/>. Acesso em: 20 nov. 2018.

CBINSIGHTS. 46 A Internet das Coisas – Fundamentos de IoT. 2015. Disponível em: <https://mva.microsoft.com/pt-br/training-courses/a-internet-das-coisas-fundamentos-de-iot-carga-horria-1h30min-12622?l=hBQb0gZSB_1205192806>. Acesso em: 05 mar. 2018.

CHEN, Y; HU, H. Internet of intelligent things and robot as a service. 2013. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1569190X12000469>. Acesso em: 11 ago. 2018.

CURRY, E. Message-Oriented Middleware. 2004. Disponível em: <https://pdfs.semanticscholar.org/98e1/90fd2ed9e15514f7f5d5ea3dbd8aeb382a9c.pdf>. Acesso em: 26 nov. 2018.

DE JESUS, C. Artificial Intelligence: What It Is and How It Really Works. Jan. 2017. Disponível em: <https://futurism.com/1-evergreen-making-sense-of-terms-deeplearning-machine-learning-and-ai//>. Acesso em: 23 set. 2017.

GUBBI, J. et al. Internet of Things (IoT): A vision, architectural elements, and future directions. 2013. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X13000241>. Acesso em: 20 set. 2018.

IBM. Disponível em: <https://www.ibm.com/>. Acesso em: 10 mar. 2018.

IBM WATSON. Disponível em: <https://www.ibm.com/watson>. Acesso em: 15 mai. 2018.

IBM WATSON VISUAL RECOGNITION SERVICE. Disponível em: <https://console.bluemix.net/docs/services/visual-recognition/index.html#about> . Acesso em: 01 jun 2018.

JSON. Disponível em: <https://www.json.org/>. Acesso em: 2 dez. 2018.

KHAREL, A. et al. Cloud Robotics using ROS. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.667.5418&rep=rep1&type=pdf>. Acesso em: 22 nov. 2018.

KASTURI, K. et al. A Review of Architecture and Applications for Internet of Things 2016. Disponível em: <https://go.galegroup.com/ps/i.do?p=AONE&sw=w&u=googlescholar&v=2.1&it=r&id=GALE%7CA466051308&sid=classroomWidget&asid=63bfbe62#>. Acesso em: 28 out. 2018.

MAINI, V. Machine Learning for Humans. Aug. 2017. Disponível em: <https://medium.com/machine-learning-for-humans/why-machine-learning-matters-6164faf1df12>. Acesso em: 1 dez. 2018.

NODE.JS. Disponível em: <https://nodejs.org/en/>. Acesso em: 18 mar. 2018.

OGLIARI, R. Popularização do padrão Publisher/Subscriber no mobile e na IoT. Aug. 2017. Disponível em: <https://imasters.com.br/desenvolvimento/popularizacao-do-padrao-publishersubscriber-no-mobile-e-na-iot>. Acesso em: 1 dez. 2018.

OPENCV. Disponível em: <https://opencv.org/>. Acesso em: 15 mai. 2018.

PAPAZOGLOU, M. Web Services SOA: Principles Technology. 2013.

REST API TUTORIAL. Disponível em: <https://restfulapi.net/>. Acesso em: 26 mar. 2018.

ROS. Disponível em: <http://www.ros.org/>. Acesso em: 1 out. 2017.

ROSBRIDGE. Version 2.0 LTS. Disponível em: <http://wiki.ros.org/rosbridge_suite>. Acesso em: 06 nov. 2018.

ROSJAVA. Disponível em: <http://wiki.ros.org/rosjava>. Acesso em: 10 mai. 2018.

QUIGLEY, M. et al. Internet of Things (IoT): A vision, architectural elements, and future directions. 2013. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X13000241>. Acesso em: 20 set. 2018.

WEST, M. An Introduction to WebSockets. 2013. Disponível em: <http://wiki.ros.org/rosjava>. Acesso em: 26 nov. 2018.

WILLOW GARAGE. Disponível em <http://www.willowgarage.com/>. Acesso em: 22 nov. 2018.