INSTITUTO FEDERAL DE SANTA CATARINA

JEFFERSON YOSHIO KAMIGASHIMA

SISTEMA PARA COLETA E DISPONIBILIZAÇÃO DE DADOS DE POTENCIAIS BURACOS

Gaspar - SC

Junho

2018

JEFFERSON YOSHIO KAMIGASHIMA

SISTEMA PARA COLETA E DISPONIBILIZAÇÃO DE DADOS DE POTENCIAIS BURACOS

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Câmpus Gaspar do Instituto Federal de Santa Catarina como requisito parcial para aprovação no Curso.

Orientador: Leonardo Ronald Perin Rauta, Me. Eng.

Gaspar

Junho

2018

K15s Kamigashima, Jefferson Yoshio
Sistema para coleta e disponibilização de dados de potenciais buracos / Jefferson Yoshio Kamigashima ; orientador, Leonardo Ronald Perin Rauta, 2018.
50 p.

Trabalho de Conclusão de Curso (graduação) – Instituto Federal de Santa Catarina, Câmpus Gaspar, Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Gaspar, 2018.

Inclui referências.

1. Buraco. 2. Mapeamento. 3. Sistema embarcado. 4. Web service. I. Rauta, Leonardo Ronald Perin. II. Instituto Federal de Santa Catarina. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. III. Título.

CDD 006

Ficha catalográfica elaborada por Cláudia Kautzmann, CRB 14/1214

RESUMO

A presença de buracos em rodovias é um problema existente em nossas estradas, agravado pela ausência de uma manutenção contínua e eficaz. Esse trabalho propõe um sistema capaz de coletar dados de possíveis buracos e disponibilizar esses dados em uma aplicação web, utilizando módulos embarcados em um automóvel e outros hospedados na nuvem. Esse sistema é composto por placas, sensores e aplicações que coletam dados de possíveis buracos, armazenam os dados localmente e posteriormente enviam esses dados até o servidor localizado na nuvem. Para acessar essas informações, um web service é disponibilizado, de forma que uma aplicação web pode consumir seus dados e apresentar um mapeamento das estradas com seus respectivos buracos. Para a validação do sistema, foram realizados testes simulando alguns cenários possíveis pelo qual o sistema pode ser submetido, com resultados que demonstram a viabilidade de empregar o sistema no desenvolvimento de planos de manutenção de rodovias e na melhoria da qualidade das nossas estradas e do conforto para os seus usuários.

Palavras-Chave: Buraco. Mapeamento. Sistema embarcado. Web service.

ABSTRACT

The presence of holes in highways is an existing problem on our roads, aggravated by the absence of continuous and effective maintenance. This work proposes a system capable of collecting data from possible holes and making them available in a web application, using modules embedded in a car and others hosted in the cloud. This system consists of boards, sensors and applications that collect data from potential holes, store them locally, and then send that data to the server located in the cloud. To access this information, a web service is made available, so that a web application can consume its data and present a map of the roads with their respective holes. For the system validation, tests were performed simulating some possible scenarios by which the system can be submitted, with results that demonstrate the feasibility of employing the system in the development of road maintenance plans and in the improvement of the quality of our roads and the comfort for their users.

Keywords: Embedded system. Hole. Mapping. Web service.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE FOTOS

LISTA DE ABREVIATURAS E SIGLAS

AVR - *Advanced Virtual RISC*
AWS - *Amazon Web Services*
CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CNT - Confederação Nacional de Transporte
DC - *Direct Current*
GPS - *Global Positioning System*
HDMI - *High-Definition Multimedia Interface*
HTTP - *HyperText Transfer Protocol*
IDE - *Integrated Development Environment*
JSON - *Java Script Object Notation*
LED - *Light Emitting Diode*
MEMS - *Microelectromechanical systems*
RCA - *Radio Corporation of America*
REST - *Representational State Transfer*
SD - *Secure Digital*
SOAP - *Simple Object Access Protocol*
URI - *Uniform Resource Identifier*
USB - *Universal Serial Bus*
V - Volt
XML - *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A malha rodoviária brasileira é responsável pelo escoamento da produção industrial e agrícola do país. Através dela é possível o tráfego de bens e pessoas pelo território nacional e a sua qualidade reflete diretamente no custo e na segurança do transporte, tornando necessária a manutenção periódica das vias para que elas ofereçam segurança e conforto para seus usuários.

No Brasil, a frota circulante de automóveis, caminhões e ônibus em 2016 contava com aproximadamente 42,9 milhões de unidades (SINDICATO..., 2016), revelando um pequeno aumento de 0,7 % em relação ao ano anterior.

De acordo com a Confederação Nacional de Transporte (CONFEDERAÇÃO..., 2016) nesse mesmo período houve, na malha rodoviária nacional, um aumento de 26,6% no número de trechos com buracos grandes, quedas de barreiras, pontes caídas e erosões, passando de 327 para 414 o número de pontos críticos. Desses, 304 são trechos com buracos com dimensões maiores que o de um pneu de um automóvel de passeio e são ocasionados geralmente pela sobrecarga dos veículos que trafegam e/ou pela espessura inadequada ou insuficiente empregadas na construção do pavimento.

No estado de Santa Catarina 59,3% (1.886 km) das rodovias apresentam algum tipo de deficiência, sendo que 43,3% da extensão avaliada apresentaram pavimento em condições regular, ruim ou péssimo, deficiência essa que eleva o custo da manutenção dos veículos e o consumo de combustível, além de reduzir a segurança no tráfego diário de automotores. Segundo a CNT (CONFEDERAÇÃO..., 2016), essas deficiências apresentadas no pavimento das rodovias fazem com que o custo operacional do transporte no Estado sofra um acréscimo estimado de 23,5%.

A manutenção das vias é de responsabilidade do governo ou das concessionárias proprietárias (CONFEDERAÇÃO..., 2016), sendo que as estatais são distribuídas nas esferas em que se encontram (municipal, estadual ou federal). Nessas, deverá acontecer um processo para a autorização do reparo e/ou manutenção, visando a obtenção de recursos para realização do serviço.

Para justificar a utilização desses recursos para manutenção, é importante que existam dados que comprovem a necessidade real do serviço com informações como localização e o tipo de dano encontrado.

Moreira *et al.* (2013) também afirmam a presença de buracos nas rodovias

brasileiras e a influência que eles exercem na ocorrência de acidentes, fato ocasionado por uma manutenção deficiente e sem um critério de escolha dos locais a serem reparados.

Trindade e Batistuta (2015) defendem a importância do monitoramento de rodovias como forma de melhorar o planejamento da manutenção e adequação das vias, coletando dados que acusaram pontos que demandam manutenção.

Com conhecimento desse problema e da importância de uma manutenção mais eficiente e melhor planejada de nossas rodovias, esse projeto visa a elaboração de um sistema que detecte a presença de buracos em rodovias, utilizando-se dos conceitos das Leis da Mecânica de Newton para identificar os possíveis buracos encontrados ao longo da estrada e realizando o seu mapeamento através do uso de GPS (*Global Positioning System*).

Essa informação, disponibilizada aos usuários, seria um recurso que auxiliaria na identificação de rodovias com trechos deficientes, possibilitando ao usuário desviar de rotas com muitos buracos e, aos responsáveis pela sua manutenção, uma ferramenta que informaria a localização dos pontos com maior necessidade de reparos.

Para embasar esse projeto, a seção 2 traz a fundamentação teórica do trabalho com os principais conceitos utilizados na resolução desse problema. A seção 3 mostra os materiais e métodos utilizados no desenvolvimento do projeto, a seção 4 apresenta os testes realizados para a validação do sistema, seguido dos resultados obtidos na seção 5. Por fim, na seção 6 são apresentadas as considerações finais e as sugestões para trabalhos futuros.

1.1 Objetivos

A seguir, são apresentados o objetivo geral e os objetivos específicos.

1.1.1 Objetivo geral

O objetivo deste trabalho é a captação e disponibilização de dados de potenciais buracos em rodovias.

1.1.2 Objetivos específicos

a) Desenvolver um sistema para detecção de potenciais buracos;
b) Desenvolver um sistema para mapeamento dos potenciais buracos;
c) Desenvolver um sistema para disponibilizar os dados coletados.

1.2 Justificativa

Devido ao fato de as empresas responsáveis pela manutenção das estradas não possuírem um critério de escolha dos locais a serem reparados (MOREIRA *et al.*, 2013), a manutenção dessas vias acaba ocorrendo sem uma frequência regular e sem um planejamento eficaz das obras a serem realizadas.

O cenário piora quando a responsabilidade dessa manutenção é de órgãos estatais, pois isso demanda a utilização de recursos públicos e, para isso, precisam ser justificados para conseguir a sua liberação. No ano de 2015, uma análise realizada pela CNT apontou que 36,52% do montante de recursos autorizados para infraestrutura de transporte rodoviário não foram utilizados (CONFEDERAÇÃO..., 2016), valores que poderiam ser convertidos em melhorias na malha viária que reduziriam o número de trechos deficientes e, consequentemente, aumentaria a segurança e satisfação de seus usuários.

Um sistema que apresente um mapeamento dos trechos com buracos existentes em rodovias e que ao mesmo tempo permite armazenar a sua localização, possibilitaria aos usuários escolher qual rota utilizar para desviar de trechos esburacados. Também seria um recurso útil para que os responsáveis pela manutenção pudessem justificar os orçamentos de manutenção da infraestrutura de transporte rodoviário ou para cobrar as empresas responsáveis, quando estas estiverem sob concessão de uma empresa de iniciativa privada.

Isso possibilitaria a implantação de planos de monitoramento planejado das vias favorecendo a qualidade da estrutura viária e um melhor aproveitamento dos recursos liberados para sua manutenção, resultando em maior segurança aos seus usuários e um menor custo ao transporte rodoviário.

Esses dados mostram a colaboração que esse sistema oportunizaria aos usuários dessas estradas e também para a economia local, justificando a relevância de sua implementação.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta os conceitos teóricos necessários ao desenvolvimento desse projeto, onde serão abordados os conceitos de estradas pavimentadas para melhor entendimento de como um buraco é formado em uma estrada; conceitos das Leis de Newton para se ter um fundamento teórico - científico de como identificar um buraco; conceito de *Representational State Transfer* (REST), princípio que será utilizado para a transmissão de dados entre aplicações; conceitos de sistemas embarcados e sensores acelerômetro, Tilt e GPS, que irão compor os equipamentos de *hardware* a serem utilizados nesse projeto.

Por fim serão apresentados os trabalhos correlatos a este projeto com suas características e uma tabela comparativa com a diferença deste aos demais.

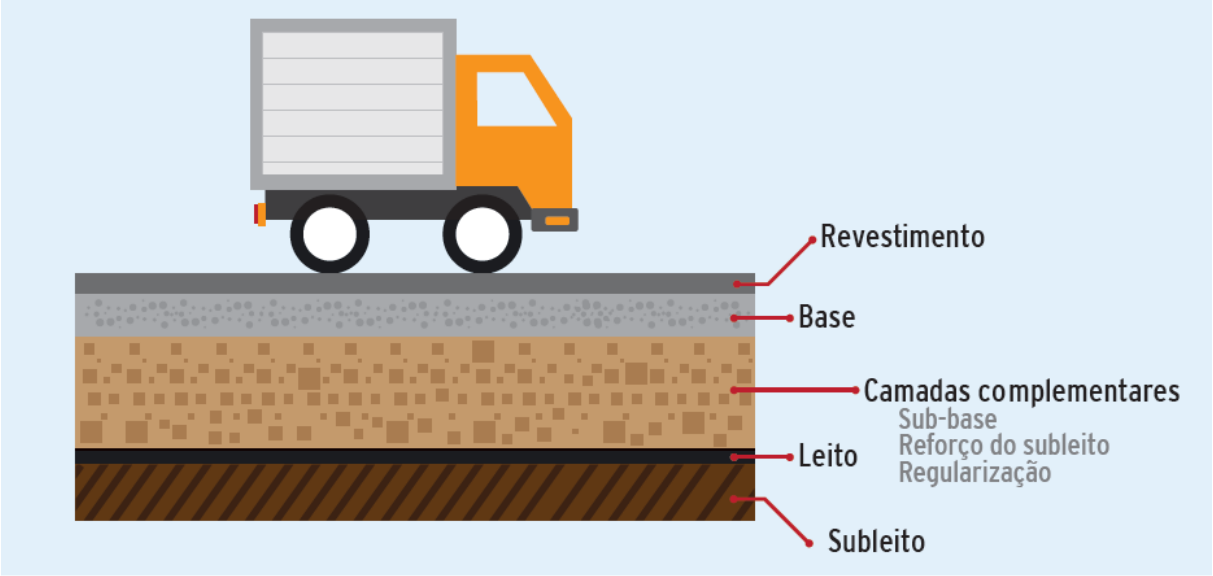
2.1 Estradas pavimentadas

As estradas pavimentadas precisam ter uma estrutura adequada para suportar o tráfego para a qual são destinadas, de modo que elas sejam seguras e resistentes, e ao mesmo tempo evitem o desgaste excessivo de pneus e a geração de ruídos (CONFEDERAÇÃO..., 2016).

Para atender essa necessidade, o processo de pavimentação ocorre por camadas de maneira que a carga exercida sobre ela seja distribuída, diminuindo a tensão e aumentando a sua durabilidade. São necessárias pelo menos duas camadas: a de revestimento, que é a parte que irá receber toda a ação do tráfego e que define a segurança e o conforto que os usuários irão dispor; e a base, que tem a finalidade de absorver a tensão recebida da camada acima.

Dependendo da estrutura do solo e do tipo de tráfego que deverá suportar, faz-se necessário a adição de camadas auxiliares: sub-base, reforço do subleito, camada de regularização e o leito. A figura 1 mostra como seria a disposição dessas camadas em uma rodovia pavimentada.

Figura 1 - Disposição das camadas do pavimento



Fonte: Confederação Nacional de Transporte, 2016.

Conforme a CNT (CONFEDERAÇÃO..., 2016, p. 37):

> • A sub-base é a camada corretiva do subleito, ou complementar à base, quando, por qualquer circunstância, não seja aconselhável construir o pavimento diretamente sobre o leito obtido pela terraplenagem.
> • O reforço do subleito é a camada executada sobre o subleito devidamente compactado e regularizado, utilizada quando se torna necessário reduzir espessuras elevadas da camada de sub-base, originadas pela baixa capacidade de suporte do subleito.
> • A camada de regularização possui espessura variável, podendo deixar de existir em alguns trechos, e possui, também, a função de corrigir falhas da camada final de terraplenagem ou de um leito antigo de estrada de terra.
> • O leito é a transição entre o terreno de fundação e o corpo do pavimento.

A conformação de todas essas camadas às necessidades do projeto de pavimentação resultará em uma via pavimentada durável e que fornecerá conforto e segurança aos seus usuários.

2.1.2 Defeitos da superfície

A ação do tempo e o fluxo contínuo dos veículos sobre a superfície do pavimento deterioram a sua estrutura, ocasionando alguns tipos comuns de danos. De acordo com a CNT (CONFEDERAÇÃO..., 2016, p. 39), “os defeitos de superfície dos pavimentos asfálticos são os danos ou as deteriorações que podem ser identificados a olho nu” e os principais tipos de defeitos que podemos encontrar na

superfície do pavimento são: trincas em malha/remendos, exsudação[1], afundamento, ondulação e buraco.

O pavimento é considerado perfeito quando não apresenta irregularidade na camada de revestimento; desgastado quando apresenta sinais de desgaste com presença de desagregação da camada asfáltica e/ou exsudação; trinca em malha/ remendos quando há trincas na malha asfáltica e/ou remendos mal executados; e destruído quando apresenta demasiada quantidade de buracos ou quando a sua camada de superfície se encontra totalmente deteriorada.

Há também a classificação de afundamento, ondulação ou buraco, quando existe a presença de um ou mais desses defeitos. Afundamentos são deformações permanentes que podem acontecer nas camadas que compõe o pavimento; ondulações são alterações transversais ao eixo da pista; e os buracos são cavidades no revestimento asfáltico.

De acordo com o Dicionário técnico: português-inglês (PANITZ, 2003, p. 51), depressão é “o segmento em que o perfil longitudinal da via tem um nível inferior ao nível médio das adjacências” e buraco é um “defeito da pista de rolamento ou do acostamento, que atinge uma ou mais camadas espessas do pavimento”.

2.2 Leis de Newton

Os estudos de Isaac Newton são a base fundamental da física que possibilitam a compreensão dos comportamentos estático e dinâmico dos corpos materiais, muitas delas publicadas na “obra Princípios matemáticos da filosofia natural” (BONJORNO *et al.*, 2013, p. 143), onde três delas, as mais conhecidas e que tratam da natureza do movimento, ficaram conhecidas com leis de Newton.

A primeira lei de Newton é conhecida como princípio da inércia e define os movimentos livres, ou seja, de quando um corpo não tem a ação de nenhuma força sobre ele. A definição original da primeira lei é que “todo corpo continua em seu estado de repouso ou de movimento uniforme em linha reta, a menos que seja forçado a mudar aquele estado por forças imprimidas sobre ele ” (OLIVEIRA *et al.*, 2013, p. 189).

O conceito dinâmico de força explica que "força é a causa que produz num

[1] Exsudação: afloramento de asfalto; defeito da superfície asfáltica de rolamento, executada com excesso de asfalto, mudando seu caráter de ligante para lubrificante podendo ocorrer deformações tipo ondulações (PANITZ, 2003, p. 153).

corpo variação de velocidade e, portanto, aceleração" (RAMALHO JUNIOR; FERRARO; SOARES, 2009, p. 198). Através dessa noção podemos entender que, segundo a primeira lei de Newton, se não houver forças agindo sobre um corpo, este se mantém como está: em movimento retilíneo uniforme ou em repouso.

A segunda lei de Newton é conhecida como princípio fundamental da Dinâmica e define o que ocorre a um corpo quando há uma força agindo sobre ele. Segundo Bonjorno *et al.* (2013, p. 147), a segunda lei diz que “a mudança do movimento é proporcional à força motriz impressa, e se faz segundo a linha reta pela qual se imprime essa força”.

Essa lei mostra que uma mesma força empregada em dois corpos de massas diferentes irá provocar uma aceleração maior no corpo de menor massa (RAMALHO JUNIOR; FERRARO; SOARES, 2009), ou seja, quanto menor a massa de um corpo, maior será a aceleração, comprovando que as grandezas massa e aceleração são inversamente proporcionais.

Isso nos permite dizer que o peso P de uma pessoa é o resultado da sua massa m multiplicada pela aceleração da gravidade g (em módulo), ou seja: P = mg. Devido à gravidade, um mesmo corpo pode ter pesos diferentes. Por exemplo: uma pessoa aqui na Terra pesa cerca de 6 vezes mais do que ela iria pesar se estivesse na Lua devido ao fato de a gravidade na Lua ser 6 vezes menor do que na Terra (BONJORNO *et al.*, 2013).

Há também a influência da resistência do ar que não deve ser desprezada. Em um corpo em queda livre existem duas forças sobre ele: a força peso e a força de resistência do ar. Segundo Bonjorno *et al.* (2013), essas duas forças têm a mesma direção vertical, porém a força peso opera para baixo e a força da resistência do ar em sentido contrário à força peso.

A presença de força de resistência do ar significa que o corpo não está em queda livre, e essa força ”depende do formato do corpo e do modo como ele se movimenta, da densidade do ar na região em que ele se movimenta e, principalmente, da velocidade desse corpo” (BONJORNO *et al.*, 2013, p. 153) sendo que quando o módulo da força da resistência do ar se se iguala ao peso do corpo em queda, este entra em uma velocidade constante de queda, chamada velocidade limite de queda.

A terceira lei de Newton é conhecida como princípio da ação e reação. Ela define que “se um corpo A aplica uma força (ação) em outro corpo B, este B aplica outra força (reação) em A, de mesma intensidade e mesma direção, mas de sentido

contrário" (OLIVEIRA, 2013, p. 196). No princípio da ação e reação as forças sempre são exercidas em corpos distintos e por isso não podem ser anuladas e qualquer uma das forças pode ser considerada a ação, sendo considerada a outra como a reação, e elas aparecem e desaparecem ao mesmo tempo.

2.3 REST

Segundo Garcia e Abilio (2017, p. 36) “REST é uma abstração dos princípios que tornam a *World Wide Web* escalável e permite oferecer serviços identificados por uma *Uniform Resource Identifier* (URI)”. Essa estrutura, usada para implementar interfaces de transmissão de dados de um domínio *HyperText Transfer Protocol* (HTTP) “sem uma camada adicional de mensagem como *Simple Object Access Protocol* (SOAP) ou *session tracking* via *cookies* HTTP” (DAL MORO; DORNELLES; REBONATTO, 2009, p. 40), não é restrita somente a sistemas que utilizem HTTP e interajam com a Web.

Para entender o funcionamento das estruturas REST, existem características comuns que devem ser respeitadas. Conforme Fielding (2000 *apud* DAL MORO; DORNELLES; REBONATTO, 2009), são elas:

- Cliente-Servidor: característica comum em aplicações Web onde existe um servidor e um cliente. O servidor disponibiliza um serviço ou conjunto de serviços e aguarda requisições destes serviços e o cliente faz a requisição de um serviço disponível no servidor, que pode tanto rejeitar como executar o serviço solicitado, e retornar uma resposta ao cliente.
- *Stateless*: a comunicação entre cliente e servidor deve ser feita sem o armazenamento de qualquer tipo de estado no servidor, exigindo que cada requisição do cliente para o servidor contenha as informações necessárias para que ela seja entendida. Nesse estilo, os estados de sessão são totalmente mantidos no cliente.
- Cache: utilizada para melhorar o desempenho, exigindo que os dados de uma resposta, vindos de uma requisição ao servidor, sejam marcados como *cacheable* ou *noncacheable* (passíveis ou não de utilização da cache), sendo que se marcada como *cacheable*, será usada para respostas posteriores de requisições equivalentes.
- Interface Uniforme: o estilo arquitetural REST enfatiza a utilização de

uma interface uniforme entre cliente e servidor, definindo quatro requisitos de interface: identificação de recursos utilizando URI (*Uniform Resource Identifier*); manipulação de recursos através de representações; mensagens auto descritivas, utilizando formatos padronizados de representação dos recursos - como JSON (*Java Script Object Notation*) e XML (e*Xtensible Markup Language*); hipermídia como mecanismo de estado da aplicação.

- Multicamada: característica que foi adicionada com objetivo de aperfeiçoar o requisito de escalabilidade da Internet.
- *Code-On-Demand*: característica opcional que permite aos clientes baixar e executar diretamente código no lado cliente, simplificando a funcionalidade da parte cliente, focando na extensibilidade, porém reduzindo a visibilidade.

Para realizar a troca de mensagens é utilizado um identificador único URI para cada recurso que poderá funcionar de maneiras diferentes de acordo com o método invocado e dos dados na requisição HTTP (DAL MORO; DORNELLES; REBONATTO, 2009).

No HTTP cada método representa uma ação que deve ser realizada sobre o recurso (GARCIA; ABILIO, 2017): GET para recuperar a representação do recurso; PUT para atualizar um recurso existente ou para criar um novo por meio de um identificador; DELETE para remover um recurso; e POST para criar um novo.

Uma característica do protocolo HTTP que é, segundo Dal Moro, Dorneles e Rebonatto (2009), muito útil na troca de mensagens REST, é o retorno de códigos de status na resposta a uma requisição, como por exemplo “HTTP/1.1 201 *Created*” quando o serviço é executado com sucesso ou um “HTTP/1.1 404 *Not Found*” quando a busca de um recurso não foi encontrada.

2.3.1 Representações

A divisão das aplicações em recursos permite uma solução dinâmica para as necessidades dos usuários de um serviço e, segundo Dal Moro, Dorneles e Rebonatto (2009, p. 43), podemos entender que:

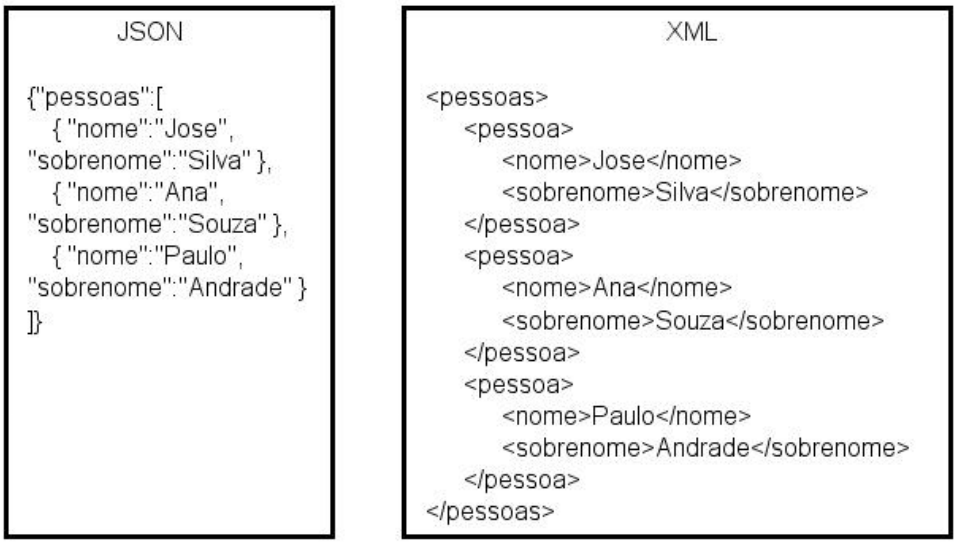
> Um recurso é uma fonte de representações, e uma representação são apenas dados sobre o estado do recurso corrente. A maioria dos recursos na Web são itens de dados próprios, como uma lista de produtos, logo uma representação óbvia de um recurso são seus dados em si. O servidor pode

> então oferecer uma lista de produtos como um documento XML, uma página Web ou em formato texto separado por vírgula.

Uma representação muito utilizada é o JSON, um formato de texto de fácil análise e geração para máquinas (SILVA, 2014) independente de idiomas e de notação conhecida pela programação. Sua estrutura pode ser constituída de uma parte com nomes e valores ou uma com lista ordenada de valores.

De acordo com a exigência REST, as estruturas do JSON são uma estrutura de dados universal, uniforme, suportado pela maioria das linguagens modernas de programação. A Figura 2 mostra um exemplo de um mesmo dado representado em uma estrutura JSON e XML.

Figura 2 - JSON x XML



Fonte: Elaborado pelo autor (2017).

2.4 Sistemas embarcados

Sistema embarcado é um sistema computacional desenvolvido para atender uma tarefa específica (MOROZ; JASINSKI; PEDRONI, 2012). São compostos por *hardware* e *software* e interagem continuamente com o ambiente a sua volta por meio de sensores e atuadores, normalmente em tempo real e sem intervenção de um usuário.

Os requisitos dos sistemas embarcados definem, conforme Moroz, Jasinski e Pedroni (2012), os componentes necessários ao sistema, fornecendo descrição dos serviços oferecidos (requisitos funcionais) e suas restrições e limitações (requisitos

não funcionais), ou seja, o que o sistema deve fazer. Os principais requisitos não funcionais que influenciam na escolha dos componentes do sistema são apresentados por Friedrich (2009 *apud* MOROZ; JASINSKI; PEDRONI, 2012):

- Limitação de recursos: como sistemas embarcados possuem recursos limitados de *hardware*, os recursos disponíveis devem ser utilizados de forma eficiente.
- Tempo real: muitas aplicações possuem requisitos temporais e necessitam que dados sejam atualizados em tempo real.
- Confiabilidade: as aplicações normalmente exigem o funcionamento contínuo por longos períodos de tempo, e o sistema deve ser confiável para garantir essa disponibilidade.
- Robustez: algumas aplicações ficam sujeitas a condições adversas de funcionamento. O sistema deve estar preparado para funcionar quando estiver fora das suas condições normais como, por exemplo, flutuações na fonte de alimentação, altas temperaturas ou vibração.
- Estabilidade: em aplicações críticas, o sistema não pode ser contaminado por uma falha isolada e deve garantir a contenção destas.
- Gerenciamento de falhas: certas aplicações precisam continuar produzindo resultados coerentes mesmo na presença de falhas, e devem prover a detecção, confinamento, recuperação e tratamento destas.
- Proteção: quando as aplicações lidam com dados sigilosos, o sistema deve evitar o acesso a estas informações sem a devida autorização.
- Segurança: algumas aplicações oferecem grandes riscos em caso de falhas. O sistema deve prevenir danos à vida, às pessoas, aos equipamentos e ao ambiente.
- Privacidade: algumas aplicações necessitam permanecerem anônimas mesmo em um ambiente público. O sistema deve isolar informações sobre si mesmo e apresentá-las somente quando for necessário.
- Escalabilidade: algumas aplicações precisam lidar com uma possível sobrecarga de trabalho e o sistema deve garantir o seu funcionamento normal nestas situações.
- Atualização: como algumas aplicações têm um prazo prolongado de

duração, o sistema deve permitir a sua atualização através da adição ou substituição de componentes do sistema.

- Consumo de energia: na maior parte das aplicações o consumo de energia é um fator crítico e deve ser minimizado ao máximo.
- Custo: todo projeto envolve muitos custos, e estes devem ser suprimidos para garantir a viabilidade do sistema.
- Manutenção: o sistema deve permitir a substituição de componentes defeituosos de *hardware* ou partes do *software*.
- Usabilidade: algumas aplicações interagem com o usuário e o sistema deve ser deve oferecer essa facilidade ao usuário.
- Portabilidade: o sistema deve permitir a migração para outra plataforma de *hardware* ou *software*.

Os sistemas embarcados podem necessitar ou não de um sistema operacional, e isso não é convencionado e varia de acordo com a experiência do desenvolvedor. Segundo Moroz, Jasinski e Pedroni (2012) alguns pontos podem ser levados em consideração nessa escolha:

- Se o sistema for modelado para atender poucas tarefas normalmente não se faz necessário o uso de um sistema operacional. De acordo com Pont (2002 *apud* MOROZ; JASINSKI; PEDRONI, 2012), o sistema operacional pode ser substituído por um *Super Loop*, que consiste em um laço infinito com todas as tarefas que devem ser executadas.
- Um sistema operacional em execução consome muitos recursos do sistema, o que aumentará os requisitos de *hardware* do projeto, aumentando o seu custo.
- O uso de um sistema operacional oferece alguns benefícios, os mais evidentes são a modularização, reutilização de código fonte e facilidade de integração de partes de *software*.
- Quando um projeto tem como requisitos alguns serviços como gerenciamento de memória por *hardware*, comunicação e sincronização de processos, tarefas com diferentes prioridades, tarefas de tempo real, protocolos de comunicação de alta complexidade - *Bluetooth*, *Universal Serial Bus* (USB), Ethernet, sistemas de arquivos ou interface gráfica com o usuário, a utilização de sistemas operacionais é recomendada.

Ao se optar por um sistema operacional, Moroz, Jasinski e Pedroni (2012) mostram dois aspectos devem ser considerados na decisão de qual sistema optar: os requisitos técnicos e os requisitos comerciais.

Em relação aos requisitos técnicos, pode-se listar:

a) Processadores suportados;
b) Requisitos de memória;
c) Desempenho;
d) Tarefas e processos;
e) Recursos e Serviços;
f) Protocolos de comunicação;
g) Ferramentas de desenvolvimento.

Em relação aos requisitos comerciais, que trata das regras de negócio, orçamento disponível e aos demais fatores não técnicos, pode-se listar:

a) Reputação;
b) Licença;
c) Custo;
d) Suporte;
e) Compatibilidade com padrões.

2.4.1 Arduino

Arduino é um tipo de sistema embarcado, sendo uma plataforma de *hardware open source* (SOUZA *et al.*, 2011) de simples utilização, que permite a interação com o ambiente através do uso de sensores. Cavalcante, Tavolaro e Molisani (2011) resumem o Arduino a uma placa de circuitos com entradas e saídas para um microcontrolador *Advanced Virtual RISC* (AVR) composto de um microprocessador, memória e periféricos de entrada/saída.

Segundo Souza *et al.* (2011), o Arduino possui uma camada simples de *software* (*bootloader*) que dispensa o uso de programadores para o chip. Ele também disponibiliza um ambiente de desenvolvimento para o computador com uma interface simples de usar e com diversas bibliotecas que permitem a comunicação com outros tipos de *hardware*.

A plataforma pode se conectar via porta USB ou serial a um computador, e quando ligada a uma fonte externa, pode alimentar circuitos auxiliares com uma saída

de tensão *Direct Current* (DC) de 3.3 Volt (V), 5V e 9V (SOUZA *et al.*, 2011).

2.4.1.1 Arduino Software (IDE)

O Arduino possui uma *Integrated Development Environment* (IDE) que contém um editor de texto para escrever código, uma área de mensagem, um console de texto, uma barra de ferramentas com botões para funções comuns e uma série de menus (ARDUINO, 2017). Ele se conecta ao *hardware* Arduino para fazer o *upload* de programas e se comunicar com eles.

Os programas escritos por ele são chamados *sketches*, cuja tradução significa esboços, e são salvos em um arquivo com a extensão ".ino". A documentação do Arduino (2017) mostra que o editor possui recursos que permitem cortar e colar textos e também a procura e substituição de textos. Ao salvar uma alteração, a área de mensagens dá *feedback* exibindo os erros.

O console exibe a saída de texto pelo *software* Arduino, apresentando os erros e outras informações. Os botões da barra de ferramentas permitem verificar e fazer *upload* de programas, criar, abrir e salvar esboços e abrir o monitor serial.

O Arduino conta com diversas bibliotecas que oferecem funcionalidades adicionais para usar nos *sketches*. Algumas delas já estão incluídas no *software* do Arduino, outros podem ser baixados ou é possível escrever a sua própria biblioteca (ARDUINO, 2017).

Para carregar um esboço, que é realizado utilizando o *bootloader* do Arduino sem a necessidade de um *hardware* adicional (ARDUINO, 2017), primeiro é necessário selecionar a porta e a placa correta para depois efetuar o *upload*. Se a placa for de uma versão mais atual do Arduino, elas serão redefinidas automaticamente e, se for uma placa mais antiga, é necessário efetuar a reinicialização na placa antes de iniciar o *upload*. Na maioria das placas, o *Light Emitting Diode* (LED) pisca enquanto o esboço é carregado e, quando o *upload* for concluído, o *software* exibirá uma mensagem informando a conclusão ou erro.

2.5 Raspberry PI

O Raspberry PI é um microcomputador completo (SOUZA; DENIS;

FERNANDES, 2015) inserido em uma placa de tamanho reduzido, possuindo um sistema operacional que deve ser instalado em um cartão *Secure Digital* (SD) em um processo de preparação e instalação relativamente simples.

Ele é composto por um processador sistema-em-um-chip de 700 MHz de 32 bits, construído sobre a arquitetura ARM11 (RICHARDSON; WALLACE, 2013) com 512 MB de memória RAM no modelo B. Conta também com duas portas USB 2.0 e não possui um disco rígido para armazenamento, para isso possui *slot* para cartão SD.

Para permitir conexão a uma rede, possui uma porta Ethernet padrão RJ45. Outra opção é utilizar um adaptador USB para conexão Wi-Fi. Uma porta *High-Definition Multimedia Interface* (HDMI) oferece saída digital de áudio e vídeo, e o modelo ainda conta com saída de áudio analógico e outra de vídeo composto padrão RCA.

A alimentação do Raspbery é feita através de um conector micro USB oferecida exclusivamente para alimentação do sistema. O sistema operacional embarcado utilizado é o Linux, e a distribuição recomendada é o Raspbian – desenvolvida com base no Debian, porém existem outras disponíveis que atendem as diversas necessidades e usos.

2.6 Acelerômetro

Um acelerômetro é um sensor eletrônico que visa medir a “grandeza física cinemática que aponta quão rapidamente a velocidade de um corpo varia ao longo do tempo” (ROCHA; MARANGHELLO; LUCCHESE, 2013, p. 103), chamada aceleração. Em um experimento desses mesmos autores foi utilizado um acelerômetro equipado com um módulo MMA 7361L triaxial, que permite a leitura simultânea da aceleração em três direções ortogonais, acoplado a um Arduino.

No funcionamento desse modelo, Rocha, Maranghello e Lucchese (2013, p. 106) explicam que:

> Nesses modelos são montadas três placas que constituem dois capacitores com uma placa central móvel [..]. Conforme se estabelece o movimento acelerado na direção considerada, a distância entre as placas se altera e a capacitância dos dois capacitores [..] varia. Ao monitorar a variação da capacitância em função da rapidez de movimentação da placa móvel, o dispositivo irá medir a aceleração em função do tempo. Um processador eletrônico monitora essas variações de capacitância e obtém a aceleração.

Logo, o acelerômetro é capaz de medir a aceleração no modo dinâmico e também a intensidade do campo gravitacional (KWAPISZ; WEISS; MOORE, 2010) no

modo estático - registrando a presença da gravidade do planeta ou sua componente em função da inclinação do eixo de leitura. As medidas dinâmicas fornecem os dados necessários para obter a aceleração. Já as medidas estáticas da direção do vetor campo gravitacional em relação a cada eixo, abre a possibilidade de medir deslocamentos angulares e até mesmo velocidades angulares com o acelerômetro trabalhando como um inclinômetro (ROCHA; MARANGHELLO; LUCCHESE, 2013).

Assim, se o corpo estiver em estática, a inclinação é encontrada lendo o sinal elétrico de saída do dispositivo. Caso contrário, o problema altera-se, devido essencialmente à mistura entre a componente gravítica (estática) e de aceleração (dinâmica) presentes no sinal.

2.7 GPS

O GPS, ou Sistema de Posicionamento Global, é utilizado para obter com precisão a posição de um objeto na Terra através do uso de satélites e receptores controlados por uma estação (TRINDADE, 2015). O sistema é composto por 24 satélites que percorrem a órbita terrestre a uma altura de 20.200 km (NASCIMENTO, 2017) efetuando, cada um, duas voltas completas por dia.

Ainda de acordo com Nascimento (2017), o sistema utiliza as informações recebidas constantemente pelos satélites para estimar a sua distância em relação a pelo menos quatro satélites utilizando a técnica chamada *Time of Arrival*. Nesse processo, a hora exata do envio desses sinais é comparada com o a hora de recebimento, e a diferença de tempo entre envio e recebimento permite ao receptor GPS determinar a sua latitude, longitude e altitude.

Segundo Dutra (2017), para realizar esse cálculo é utilizada uma equação que leva em consideração a constante da velocidade da luz multiplicada pelo tempo que o sinal levou para percorrer o trajeto do satélite até o receptor, resultando na distância entre esses dois pontos. Com essa distância, o equipamento obtém a posição geográfica através de cálculos de trilateração envolvendo conceitos de geometria esférica, porém um satélite é capaz de obter apenas a posição em algum lugar na superfície terrestre, daí a necessidade de pelo menos quatro satélites.

Utilizando o dado do segundo satélite, forma-se uma área circular com a intersecção das duas posições. Um terceiro satélite restringe a posição a dois pontos, e o quarto é utilizado para determinar um ponto, que é a localização geográfica do

receptor.

Mesmo com a utilização desses cálculos, Dutra (2017) diz que fatores como a órbita do satélite, interferência atmosférica e erros nos relógios do receptor e do satélite interferem na precisão do posicionamento, e Nascimento (2017) completa essa informação mostrando que os dispositivos GPS possuem uma margem de erro de 5m a 30m.

2.8 Trabalhos correlatos

Durante o levantamento bibliográfico para produção da fundamentação teórica, foram realizadas pesquisas no portal de periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e no Google Acadêmico, onde foram encontrados os trabalhos de Borges *et al.* (2011) e de Moreira *et al.* (2013).

Borges *et al.* (2011) apresentam a proposta de um sistema para identificar a localização geográfica dos buracos em estradas. No seu experimento foi utilizado um sistema com um microcontrolador, um módulo *Bluetooth*, um aparelho celular com GPS e um Wii *remote* adaptados a um carro de controle remoto. Com o projeto, Borges *et al.* (2011) concluíram que o sistema se mostrou viável para a detecção de buracos em estradas.

No outro projeto, Moreira *et al.* (2013) desenvolveram um sistema semelhante. O objetivo do projeto era detectar buracos em estradas ou rodovias utilizando um acelerômetro. Para tanto foram utilizados um microcontrolador, um acelerômetro e um computador com um *software* para interpretar os dados processados. No experimento foi utilizado uma bicicleta para simular os testes. O resultado obtido por Moreira *et al.* foi um sistema capaz de identificar todos os buracos pelo qual o protótipo passou, com a ressalva de que ao sair de um buraco o sistema acabava identificando outro buraco.

Em ambos os projetos a ideia de utilizar um sistema embarcado para identificar buracos obteve resultados satisfatórios, porém com suas aplicações em protótipos que não refletem na totalidade o cenário real. Esse fator acaba influenciando os resultados, pois a disposição dos eixos, o peso e as dimensões maiores que um carro possui modificaria a lógica necessária para a identificação de um buraco.

No trabalho de Borges *et al.* (2011) o protótipo utilizado tinha dimensões muito distantes de um carro em tamanho real, e a dimensão dos buracos pelos quais ele passou também eram igualmente menores. Esse detalhe foi explorado no trabalho de

Moreira *et al.* (2013) onde o protótipo pôde passar por buracos no tamanho real pelo qual um carro passaria. Entretanto, o protótipo ainda estava longe da estrutura de um carro, uma vez que o carro possui quatro eixos e um peso substancialmente maior.

O intuito desse projeto é aplicar um sistema semelhante aos citados anteriormente em um carro, simulando o ambiente real pelo qual o sistema iria passar. Para isso, as contribuições dos estudos anteriores serão de grande valia, fornecendo um norte a ser seguido e incrementando o nível de informações necessárias para desenvolver um projeto que consiga obter o resultado esperado.

O Quadro 1 apresenta um comparativo entre as principais diferenças entre os projetos citados em relação a este projeto.

Quadro 1 - Comparativo entre os trabalhos correlatos

Trabalho	**Localização via GPS?**	**Buracos em tamanho real?**	**Protótipo em escala real?**	**Disponibilização dos dados**
Borges *et al.*	Sim	Não	Não	Não
Moreira *et al.*	Não	Sim	Não	Não
Este trabalho	Sim	Sim	Sim	Sim

Fonte: Elaborado pelo autor (2017).

3 MATERIAIS E MÉTODOS

Essa seção descreve os materiais e métodos que foram utilizados no desenvolvimento deste projeto e a modelagem do sistema. Para elaboração deste projeto foi realizada uma pesquisa exploratória sobre todos os conceitos necessários para o desenvolvimento de um sistema que possibilite a identificação e localização de possíveis buracos em rodovias.

3.1 Solução desenvolvida

A solução proposta é a coleta de informação dos dados de possíveis buracos encontrados em rodovias, apresentando-os na forma de um mapeamento de buracos em rodovias. Para isso, o sistema desenvolvido possui um módulo embarcado que coleta informações de ocorrências de possíveis buracos e disponibiliza essa informação através de uma aplicação web instalada na nuvem. Assim, o sistema faz uso dos seguintes equipamentos de hardware:

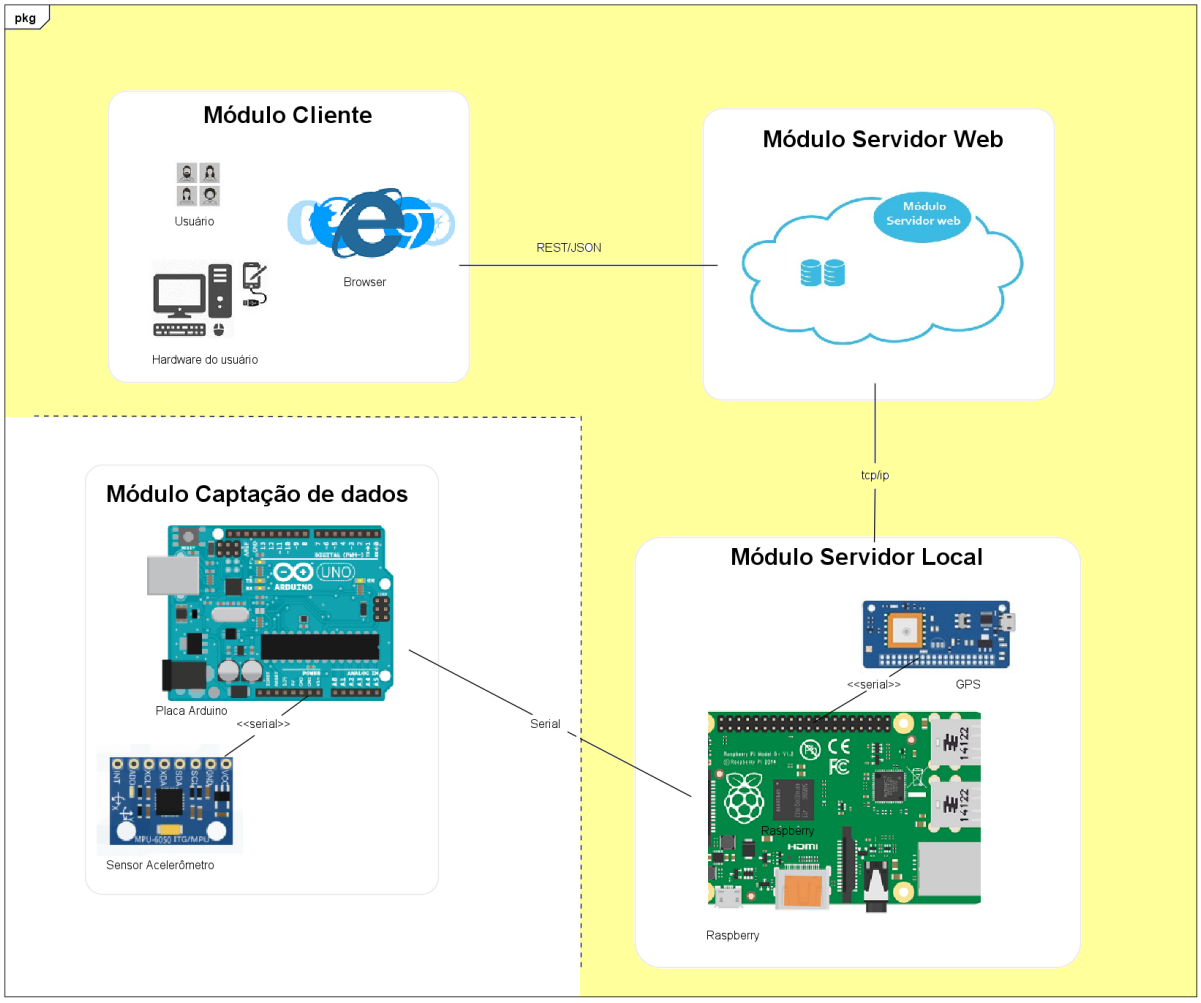
a) 1 Placa Arduino UNO;
b) 1 Sensor acelerômetro MPU6050;
c) 1 Raspberry PI3;
d) 1 Sensor GPS GY-NEO6Mv2 + antena cerâmica;
e) 1 Computador.

Em relação ao *software*, para obtermos as funcionalidades necessárias para o funcionamento do sistema, foram desenvolvidos os seguintes módulos: módulo servidor web; módulo cliente; módulo servidor local e módulo captação de dados. Para o funcionamento do sistema completo, os módulos interagirão entre si até os dados chegarem a um usuário conectado ao módulo cliente.

A Figura 3 apresenta o diagrama de distribuição, exemplificando as interações entre os módulos até as informações chegarem ao cliente. A parte destacada em branco, dentro da linha tracejada, não foi completamente desenvolvida nesse projeto devido a proporção que o projeto tomou. Essa decisão de limitar o escopo do projeto foi necessária para que fosse possível, ao final deste projeto, entregar um sistema capaz de cumprir os objetivos delineados, pois cada um dos módulos utiliza tecnologias diferentes e necessitam que a integração entre eles funcione sem restrições, por essa razão foi dado ênfase na integração de todos os módulos, e

retirado do projeto a instalação dos módulos captação de dados nos quatro eixos do veículo.

Figura 3 - Diagrama de distribuição



Fonte: Elaborado pelo autor (2018).

O módulo de captação de dados fará uso de uma placa Arduino instalada em cada um dos eixos do carro, próximo ao amortecedor. Nessa placa será acoplado o sensor acelerômetro, que fará a captação dos valores das forças gravitacionais exemplificadas na seção 2.2, que serão recebidos pela aplicação instalada no Arduino e enviados ao servidor local.

O módulo servidor local deverá integrar com o módulo de captação para receber os dados dos sensores. Como nesse projeto o módulo de captação não foi completamente desenvolvido, foi integrado uma placa Arduino, com um sensor acelerômetro instalado, ao Raspberry através de comunicação USB, que possibilita o

recebimento da leitura do acelerômetro e a alimentação do Arduino.

Além de integrar com o Arduino, nesse módulo são armazenados os dados dos eventos capturados pelos sensores e, no momento em que o usuário solicitar o envio, existindo a disponibilidade de uma conexão com a internet, o módulo servidor local deve integrar com o módulo servidor web, permitindo o envio dos dados recolhidos pela aplicação instalada no Raspberry para armazenamento no banco de dados do servidor web, utilizando uma aplicação cliente que realiza solicitações para a aplicação servidora do módulo servidor web.

O módulo Servidor web precisa estar disponível para o módulo servidor local enviar os dados e para o módulo cliente consumir esses dados. Para armazenar esses dados, foi utilizado um banco de dados não relacional MongoDB, que possui alta escalabilidade e facilidade na integração com outras tecnologias, hospedado na nuvem através de um serviço de gerenciamento de banco de dados, o mLab[2]. A inserção nesse banco de dados ocorre via socket, através de uma aplicação servidora disponível na web, implementada utilizando TCP/IP, que recebe requisições da aplicação cliente existente no módulo servidor local.

Para disponibilizar esses dados para o módulo cliente, foi implementado um *web service* RESTfull utilizando o framework Jersey e o servidor Apache Tomcat, que disponibiliza consultas ao banco de dados com a possibilidade de efetuar buscas por data e/ou carro.

Por fim, para disponibilizar e apresentar os dados para o usuário, foi implementado uma página web que permite buscar todos os buracos encontrados ou realizar uma busca filtrada por período, por nome do carro ou por carro em um determinado período. Esses dados são consumidos do *web service* e apresentados em um mapa utilizando a API do *Google Maps* e também em uma tabela.

No mapa, cada buraco é representado por um marcador que, ao clicar sobre ele com o mouse, apresenta um quadro com as informações individuais de cada buraco. O site é disponibilizado para o usuário nesse momento somente em máquina local, utilizando o servidor Apache e o endereço do *localhost*.

As dificuldades encontradas ocorreram na integração entre os módulos, mais especificamente na aplicação web, onde houve incompatibilidade de dados no recebimento das informações do *web service*, sendo necessário tratamentos para que

[2] https://mlab.com

os dados fossem armazenados e apresentados corretamente; e também, na captação dos valores dos sensores no módulo servidor local, onde foram necessários vários tratamentos de dados e configurações diretamente no sistema operacional do Raspberry.

3.2 Requisitos funcionais e não funcionais

O sistema apresenta os seguintes requisitos funcionais:

a) Captar informação do sensor acelerômetro;
b) Captar informação do sensor GPS;
c) Enviar informações dos sensores para a aplicação responsável;
d) Identificar a localização do momento em que for identificado um possível buraco;
e) Armazenar os dados no servidor local (Raspberry);
f) Armazenar os dados no servidor web;

Requisitos não funcionais:

a) Utilizar Rest para comunicação entre o servidor web e o módulo cliente;
b) Utilizar sistema Linux no Raspberry;
c) Enviar os dados via comunicação *wireless* para o servidor web;
d) Utilizar JSON no serviço Rest.

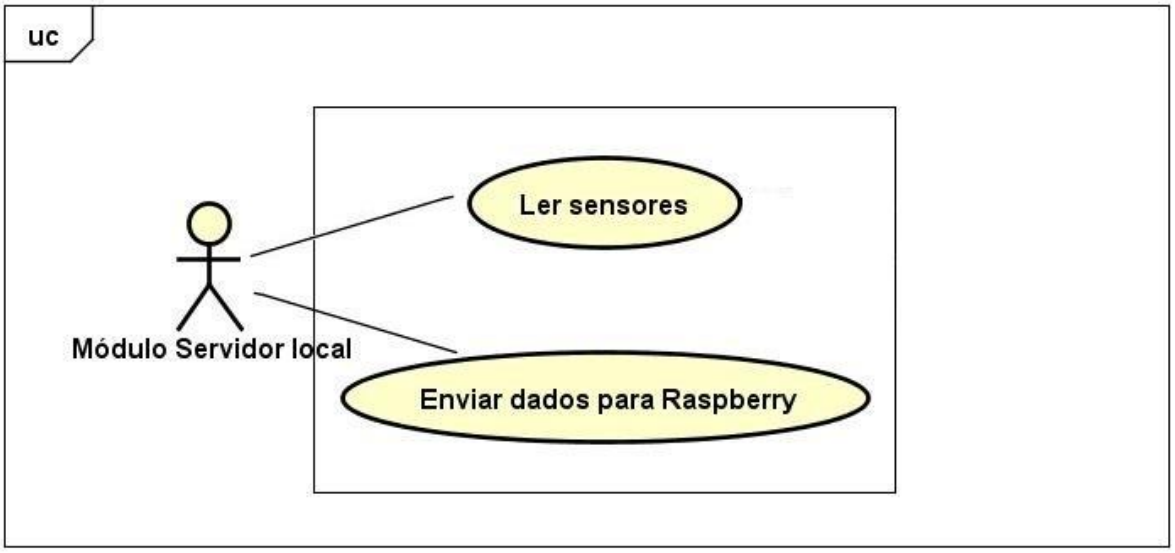
3.3 Diagramas de casos de uso

Nessa seção são apresentados os diagramas de casos de uso, identificando o comportamento de cada módulo, e os respectivos quadros com a descrição de cada caso.

3.3.1 Casos de uso do módulo captação

Nesse módulo, teremos os comportamentos de ler sensores e enviar dados para o Raspberry, conforme demonstrado na Figura 4.

Figura 4 - Módulo captação



Fonte: Elaborado pelo autor (2017).

Os Quadros 2 e 3 descrevem os casos de uso do módulo captação.

Quadro 2 - Ler sensor

Caso de uso: Ler sensor	
Ator principal	Módulo servidor local
Ator secundário	-
Resumo	Este caso de uso mostra as etapas percorridas para se obter os valores captados pelo sensor
Pré-condições	Sensor conectado
Pós-condições	-
Fluxo normal	
Ações do ator	**Ações do sistema**
	Define os pinos de entrada e saída o Arduino
	Efetua a leitura dos valores do sensore
	Armazena os valores em uma variável String
	➔ Caso de uso: Enviar informações para Raspberry

Fonte: Elaborado pelo autor (2017).

Quadro 3 - Enviar informações para Raspberry

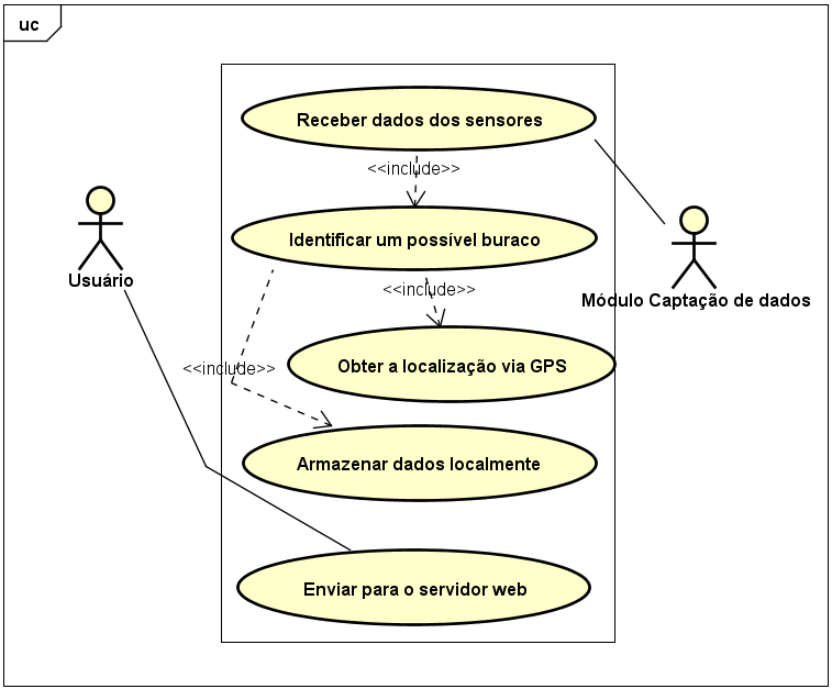
Caso de uso: Enviar informações para Raspberry	
Ator principal	Módulo servidor local
Ator secundário	-
Resumo	Este caso de uso mostra as etapas percorridas para enviar os valores captados pelo sensor ao módulo servidor local
Pré-condições	Comunicação com a porta serial / Comunicação com aplicação do Raspberry
Pós-condições	-
Fluxo normal	
Ações do ator	**Ações do sistema**
	➔ Caso de uso: Ler sensor
	Envia os dados do sensor para a aplicação do Rasperry

Fonte: Elaborado pelo autor (2017).

3.3.2 Casos de uso módulo servidor local

Conforme apresentado na Figura 5, esse módulo possui os comportamentos de receber dados do Arduino, identificar um possível buraco, obter a localização do carro, armazenar os dados localmente e enviar para a web.

Figura 5 - Módulo Servidor local



Fonte: Elaborado pelo autor (2017).

Os Quadros 4, 5, 6, 7 e 8 descrevem os casos de uso do módulo servidor local.

Quadro 4 - Receber dados do Arduino

Caso de uso: Receber dados dos sensores	
Ator principal	Módulo Captação de dados
Ator secundário	-
Resumo	Este caso de uso mostra as etapas percorridas para receber os valores captados pelos sensores do sistema
Pré-condições	Comunicação com a porta serial
Pós-condições	-
Fluxo normal	
Ações do ator	**Ações do sistema**
	Define a porta para efetuar a comunicação
Envia os dados	Recebe os dados da aplicação do Arduino
	Efetua tratamento dos valores obtidos do sensor acelerômetro
	Armazena as 10 últimas leituras do sensor
	Caso de uso: Identificar um possível buraco
Fluxo alternativo	
Ações do ator	**Ações do sistema**
	Porta definida não existe
	Reinicia o processo

Fonte: Elaborado pelo autor (2017).

Quadro 5 - Comparativo entre os trabalhos correlatos

Caso de uso: Identificar um possível buraco	
Ator principal	-
Ator secundário	-
Resumo	Este caso de uso mostra as etapas percorridas para identificar um possível buraco
Pré-condições	Receber dados dos sensores
Pós-condições	-
Fluxo normal	
Ações do ator	**Ações do sistema**
	Analisa o valor da leitura atual do acelerômetro
	Algoritmo identifica um buraco
	Armazena as 10 próximas leituras do sensor acelerômetro
	Busca a posição do carro
	Caso de uso: Obter a localização via GPS
Fluxo alternativo	
Ações do ator	**Ações do sistema**
	Algoritmo não identifica um buraco
	Analisa a próxima leitura do acelerômetro

Fonte: Elaborado pelo autor (2017).

Quadro 6 - Obter a localização via GPS

Caso de uso: Obter a localização via GPS	
Ator principal	-
Ator secundário	-
Resumo	Este caso de uso mostra as etapas percorridas para obter a localização do carro através do sensor GPS
Pré-condições	Identificação de um buraco
Pós-condições	-
Fluxo normal	
Ações do ator	**Ações do sistema**
	Obtém a leitura atual do sensor GPS
	Efetua tratamento dos dados
	Caso de uso: Armazenar dados localmente
Fluxo alternativo	
Ações do ator	**Ações do sistema**
	Erro ao obter algum dado do GPS
	É lançada uma exceção
	Grava essa informação em branco ou com valor 0 (zero)

Fonte: Elaborado pelo autor (2017).

Quadro 7 - Armazenar dados localmente

Caso de uso: Armazenar dados localmente	
Ator principal	-
Ator secundário	-
Resumo	Este caso de uso mostra as etapas percorridas para armazenar os dados do buraco identificado
Pré-condições	Identificação de um buraco
Pós-condições	-
Fluxo normal	
Ações do ator	**Ações do sistema**
	Converte os valores tratados após a identificação de um buraco em objeto JSON
	Grava a informação do objeto JSON em arquivo
Fluxo alternativo	
Ações do ator	**Ações do sistema**
	Erro ao tentar gravar em arquivo
	É lançada uma exceção
	Continua o processo

Fonte: Elaborado pelo autor (2017).

Quadro 8 - Enviar para servidor web

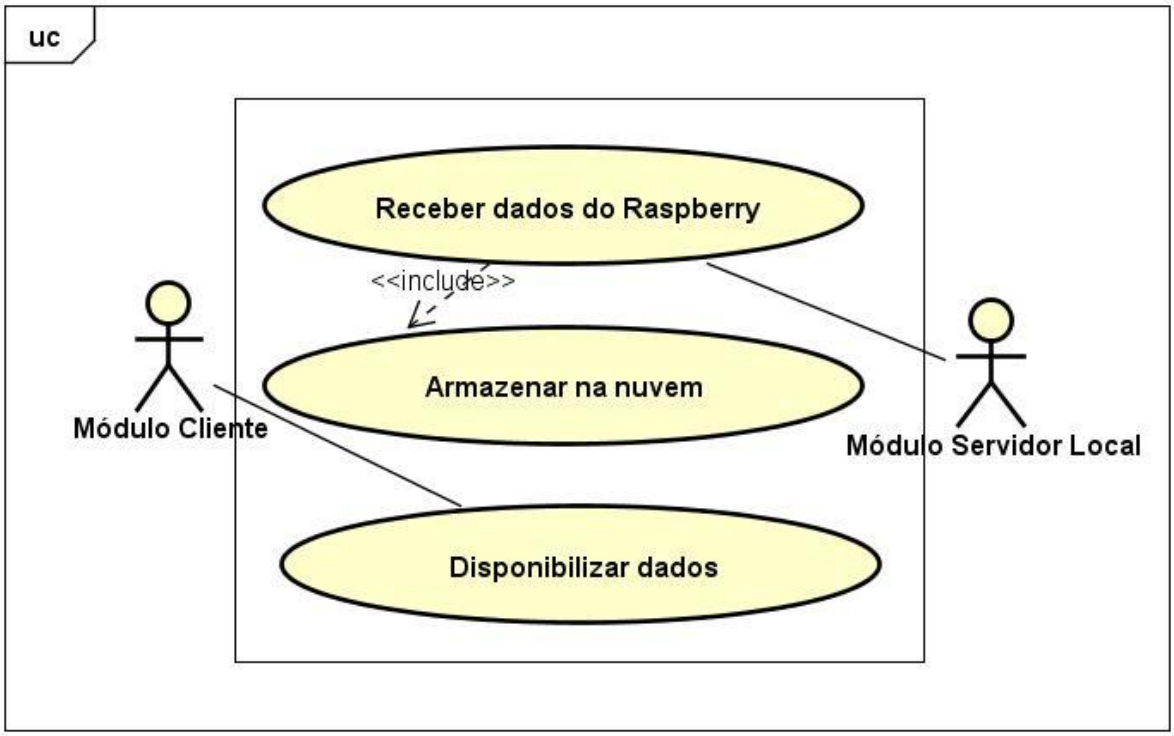
Caso de uso: Enviar para servidor web	
Ator principal	Usuário
Ator secundário	-
Resumo	Este caso de uso mostra as etapas percorridas para enviar os dados do buraco identificado para o servidor web
Pré-condições	Conexão wireless / Dados em arquivo
Pós-condições	-
Fluxo normal	
Ações do ator	**Ações do sistema**
Inicia a aplicação	Estabelece conexão com o servidor web
	Acessa os dados em arquivo
	Envia os arquivos
	Apaga os arquivos
	Encerra conexão com o servidor web
Fluxo alternativo	
Ações do ator	**Ações do sistema**
	Erro ao tentar acessar o arquivo
	É lançada uma exceção
	Reiniciado o processo
Fluxo alternativo	
Ações do ator	**Ações do sistema**
	Erro ao conectar com o servidor web
	É lançada uma exceção
	Efetua nova tentativa de conexão
Fluxo alternativo	
Ações do ator	**Ações do sistema**
	Não foi possível armazenar o dado no servidor web
	Arquivo não enviado é armazenado em uma pasta separada
	Efetua o envio do próximo arquivo

Fonte: Elaborado pelo autor (2017).

3.3.3 Casos de uso módulo servidor web

A Figura 6 apresenta os comportamentos do módulo servidor web: receber dados do Raspberry, armazenar na nuvem e disponibilizar dados.

Figura 6 - Módulo servidor web



Fonte: Elaborado pelo autor (2017).

Os Quadros 9, 10 e 11 descrevem os casos de uso do módulo servidor web.

Quadro 9 - Receber dados do Raspberry

(Continua)

Caso de uso: Receber dados do Raspberry	
Ator principal	Módulo servidor local
Ator secundário	-
Resumo	Este caso de uso mostra as etapas percorridas para receber os dados enviados do Raspberry
Pré-condições	Conexão entre servidor local e servidor web
Pós-condições	-
Fluxo normal	
Ações do ator	**Ações do sistema**
	Estabelece conexão de servidor
Envia dados	Recebe os dados via TCP/IP
	Carrega os dados em um objeto Buraco
	➔ Caso de uso: Armazenar na nuvem
Fluxo alternativo	
Ações do ator	**Ações do sistema**
	Erro ao tentar acessar o banco de dados
	É lançada uma exceção
	Reiniciado o processo

Quadro 9 - Receber dados do Raspberry

(Conclusão)

Fluxo alternativo	
Ações do ator	**Ações do sistema**
	Erro ao conectar com o servidor local
	É lançada uma exceção
	Aguarda próxima requisição do servidor local

Fonte: Elaborado pelo autor (2017).

Quadro 10 - Armazenar na nuvem

Caso de uso: Armazenar na nuvem	
Ator principal	-
Ator secundário	-
Resumo	Este caso de uso mostra as etapas percorridas para armazenar os dados em um banco de dados na nuvem
Pré-condições	-
Pós-condições	-
Fluxo normal	
Ações do ator	**Ações do sistema**
	Acessa o banco de dados
	Grava os dados do objeto Buraco em banco
Fluxo alternativo	
Ações do ator	**Ações do sistema**
	Erro ao tentar acessar o banco de dados
	Reinicia o processo

Fonte: Elaborado pelo autor (2017).

Quadro 11 - Disponibilizar dados

(Continua)

Caso de uso: Disponibilizar dados	
Ator principal	Módulo Cliente
Ator secundário	-
Resumo	Este caso de uso mostra as etapas para disponibilizar os dados dos buracos para o cliente
Pré-condições	-
Pós-condições	-
Fluxo normal	
Ações do ator	**Ações do sistema**
	Mantém comunicação de servidor
	Disponibiliza acesso às informações via *web service*
Faz solicitação	Envia dados

Quadro 11 - Disponibilizar dados

(Conclusão)

Fluxo alternativo	
Ações do ator	**Ações do sistema**
	Erro ao estabelecer a comunicação
	É lançada uma exceção
	Reiniciado o processo

Fonte: Elaborado pelo autor (2017).

3.4 Validação

Para a validação do sistema foi realizada uma pesquisa experimental com os módulos desenvolvidos onde foram realizados testes para validar a funcionalidade do módulo servidor local, módulo servidor web e módulo cliente.

Foram realizados 9 testes, (I) teste em funcionamento no carro por 15 minutos; (II) teste em funcionamento no carro com interrupção; (III) teste em funcionamento no carro configurado para encontrar buracos em um intervalo menor de tempo; (IV) teste em funcionamento no carro configurado para encontrar efetivamente um buraco; (V) teste de envio dos dados; (VI) teste de inserção no banco de dados; (VII) teste de persistência e armazenamento de dados; (VIII) teste dos filtros de consulta e da apresentação dos dados; (IX) teste do sistema funcionando por completo.

No módulo servidor local, foram realizados testes para verificar a captura das leituras do acelerômetro e do GPS e o envio desses dados para o servidor web. Para testar a captura das leituras dos sensores, foi instalado um programa configurado para que, a cada 15 segundos, simulasse que o sistema encontrou um buraco. Guardando assim as informações da posição do carro através do GPS e também a leitura atual, as 10 leituras anteriores e as 10 leituras posteriores à identificação do buraco, durante um período de 15 minutos.

Nos testes realizados no automóvel, o sistema foi ligado à alimentação 12V. Devido à alimentação das tomadas do carro serem energizadas no acionamento do motor, cada vez que o carro era desligado, essa alimentação era cortada automaticamente, interrompendo assim o funcionamento da aplicação. A foto 1 mostra o funcionamento do sistema no interior do carro. Foi permitido que o sistema funcionasse por um tempo menor para testar a persistência de dados em condições adversas de funcionamento como, por exemplo, quando o motor do carro desliga sem a intenção do condutor.

Foto 1 - Instalação do sistema no interior do carro para testes

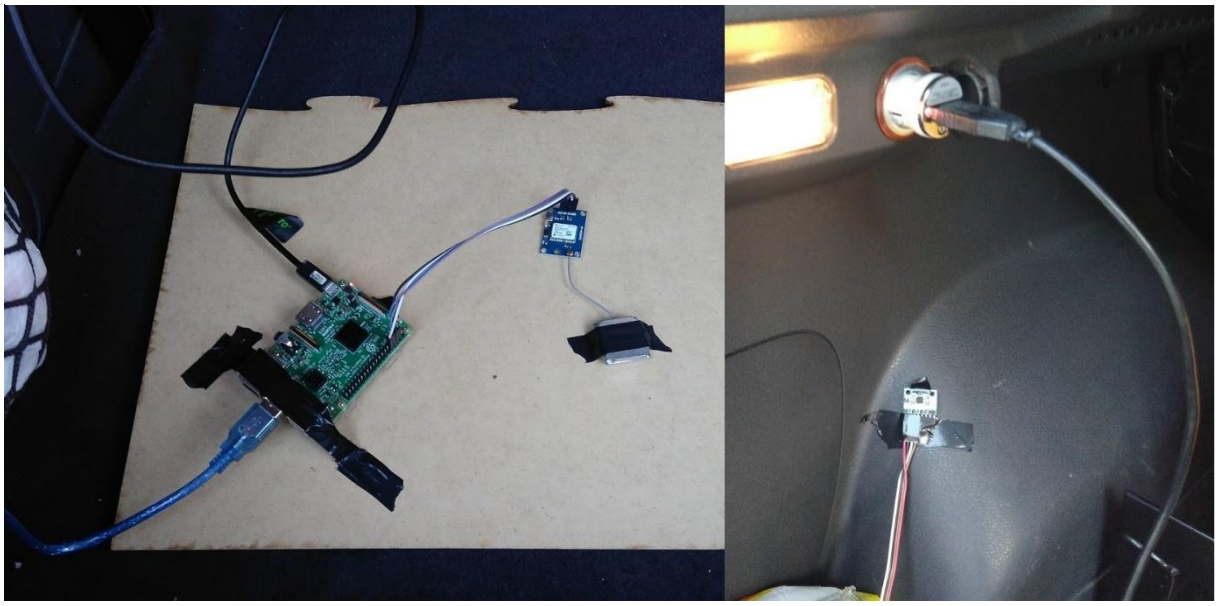
Fonte: Elaborado pelo autor (2018).

Nos primeiros testes, após configurada a aplicação no Raspberry, o carro percorreu dois trajetos diferentes: no teste (I) pelo tempo necessário para o sistema rodar o programa por completo (15 minutos), e no teste (II) o sistema funcionou por aproximadamente 7 minutos até ser interrompido o seu funcionamento pelo desligamento do motor do carro. Depois, no retorno desse trajeto, o sistema foi religado e pode funcionar pelos 15 minutos necessários para a aplicação rodar por completo, totalizando 22 minutos de captura de buracos no segundo teste.

O teste (III) foi realizado alterando a configuração do sistema para simular o encontro de um buraco a cada 7 segundos durante o período de 7,5 minutos. Após percorrer o trajeto por aproximadamente 3 minutos, o motor do carro foi desligado e logo em seguida religado, permitindo que a aplicação voltasse a funcionar, agora pelos 7,5 minutos necessários. O sistema foi submetido a esse teste para verificar a disponibilidade da aplicação após uma interrupção repentina da alimentação, e para verificar a capacidade de buracos/segundo que ele poderia capturar.

Após finalizar os testes descritos acima, a configuração foi alterada para encontrar efetivamente um buraco, sendo esse o teste (IV) realizado. Como o algoritmo para identificar um buraco não foi desenvolvido nesse projeto, foi convencionado que o sistema consideraria como buraco os eventos dos sensores onde a leitura “AngleX” do acelerômetro ultrapassasse o valor de 16500. Esse foi um valor aleatório escolhido para que a aplicação não ficasse sensível a ponto de encontrar eventos a todo instante, nem rigoroso demais impedindo que fosse encontrado algum evento. O sistema então foi instalado no porta-malas do carro, onde o acelerômetro foi fixado sobre a caixa de ar do eixo traseiro esquerdo, com o restante do sistema fixado sobre uma prancha de madeira, alimentado pela saída 12V do automóvel (foto 2).

Foto 2 – Sistema instalado no porta-malas do automóvel



Fonte: Elaborado pelo autor (2018).

Nesse quarto teste, o sistema foi programado para rodar durante um período de 10 minutos, no qual, durante o trajeto percorrido aleatoriamente, o carro fez várias paradas e, em cada uma, o carro foi desligado, fazendo com que o sistema não funcionasse pelo tempo total de 10 minutos, simulando a condição de uso na rotina diária de uma pessoa que precisa realizar vários pequenos trajetos, como fazer compras, abastecer e levar o filho para escola.

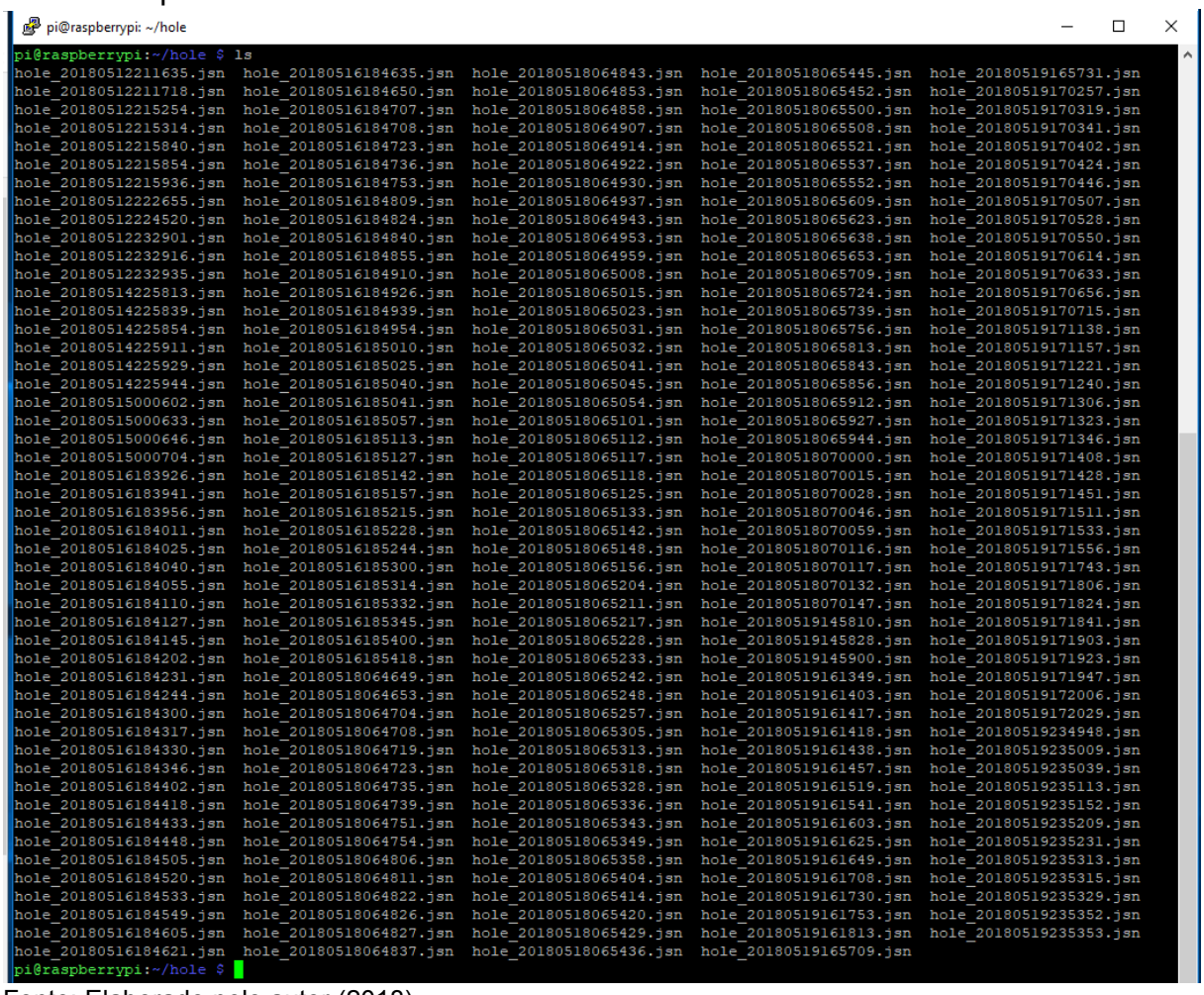
Com os dados dos buracos armazenados no servidor local, foi possível realizar o teste (V): o envio de dados para o servidor web. Em cada vez que foi verificado o armazenamento dos dados coletados no Raspberry, a aplicação de envio era iniciada

para integrar os dados coletados com o servidor web.

O teste (VI) foi realizado para confirmar se os dados foram realmente inseridos, possibilitando a sua consulta. Para isso, foi acessado o gerenciador do banco de dados e, após, foi acessado a página clicando para buscar todos os buracos.

Já o teste (VII), de persistência e de armazenamento de dados, também foi realizado no servidor web: foram incluídos na pasta em que são armazenados os buracos no Raspberry, todos os arquivos JSON de buracos encontrados (244 no total) e então iniciou-se a aplicação de envio. A foto 3 apresenta a pasta no Raspberry onde ficam armazenados os arquivos JSON com os dados de cada buraco encontrado.

Foto 3 - Arquivos armazenados no servidor local



Fonte: Elaborado pelo autor (2018).

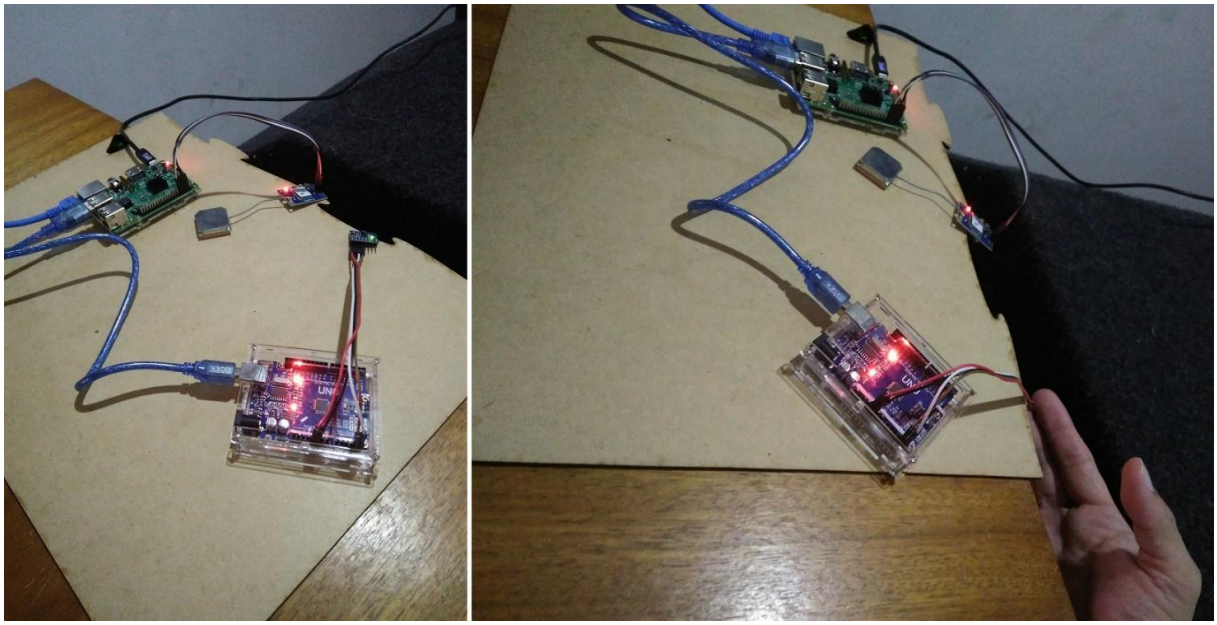
Após a inserção no banco de dados, o teste (VIII) foi realizado acessando a página web e efetuando consulta a todos os buracos (sem filtro), e depois foram realizadas consultas com o filtro de nome do carro, filtro por período e por último

consulta com o filtro de nome do carro dentro de um período.

Como teste final (IX), foi implementado um programa com as configurações do teste que encontrou efetivamente um buraco. Esse teste não foi realizado dentro do veículo, foi realizado próximo ao computador por um período de 75 segundos. O objetivo era simular o sistema funcionando integralmente, desde a identificação do buraco até a apresentação dos dados na página web.

Nesse teste (IX) o sistema foi posicionado com o sensor acelerômetro na extremidade de uma mesa para que, no momento oportuno, fosse possível incliná-lo para baixo, alterando assim o valor do “AngleX” do acelerômetro (foto 4).

Foto 4 – Teste final



Fonte: Elaborado pelo autor (2018).

4 RESULTADOS

Com os testes realizados, foi possível verificar que no teste (1), após percorrer os trajetos e acessado o sistema operacional do Raspberry, a aplicação foi capaz de encontrar 60 buracos e 95 buracos no teste (2). No teste (3), onde o sistema estava configurado para encontrar buracos em um intervalo menor de tempo, a aplicação foi capaz de encontrar 37 buracos.

Analisando o resultado dos três primeiros testes, verificou-se que há uma limitação do sistema para encontrar buracos em períodos menores que 10 segundos.

A limitação ocorre pelo fato de que, após identificar um possível buraco, o sistema é programado para guardar as 10 próximas leituras do acelerômetro, que envia essa informação para o Raspberry a cada segundo. Como nesse momento ele está aguardando receber as 10 próximas leituras para gravar as informações em arquivo, foi colocada uma trava no sistema que só é liberada quando terminar a gravação para garantir a integridade dos dados coletados.

Essa funcionalidade de guardar as leituras seguintes foi inserida no sistema para que ela possa ser utilizada no desenvolvimento do algoritmo de identificação de buracos, sendo que, após esse algoritmo ser desenvolvido, existe a possibilidade de se retirar essa funcionalidade, permitindo reduzir o intervalo desse bloqueio.

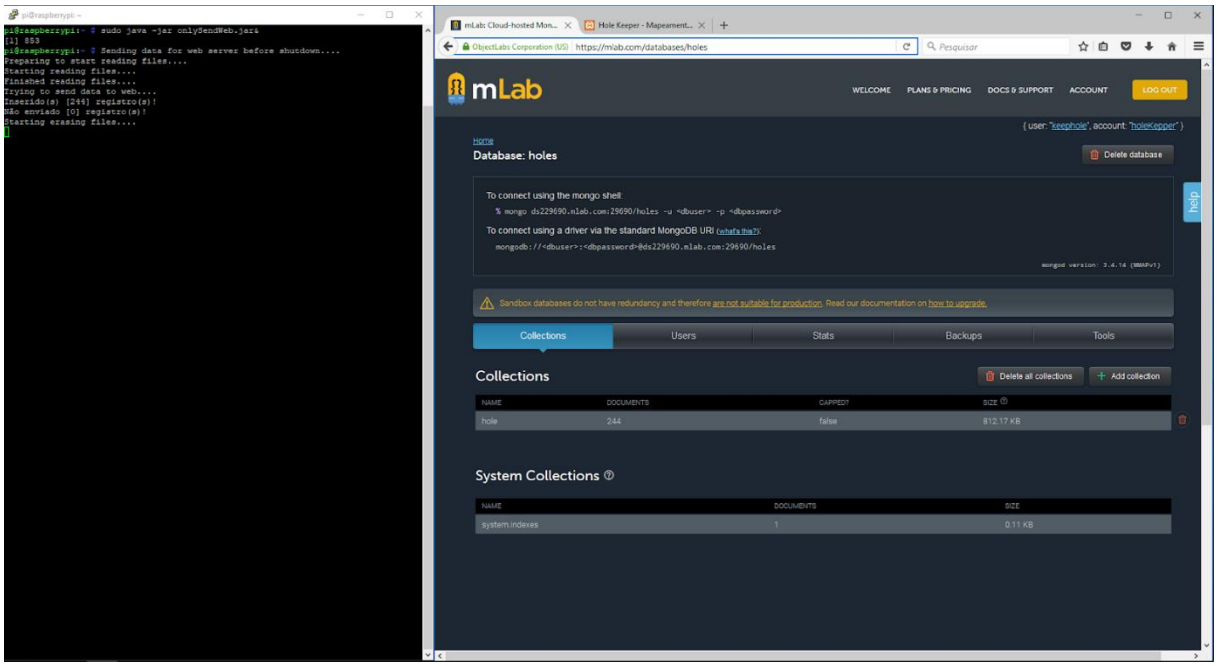
O teste (4) considerava um buraco quando os eventos captados pelos sensores tivessem a leitura do “AngleX” com valor acima de 16.500 e, com essa configuração, no trajeto percorrido, foram identificados 12 buracos. Demonstrando assim que é possível o desenvolvimento de um algoritmo para identificação de buracos utilizando as variações de forças obtidas através do acelerômetro.

Nos testes (5) e (6), como resultado do envio dos dados coletados ao servidor web, no teste (1) o servidor retornou que foram inseridos 60 buracos, no envio dos buracos do teste (2) foram 95 buracos, e no trajeto aleatório do teste (3), 12 registros foram inseridos no banco de dados.

Para confirmar se os dados foram realmente inseridos, foi acessado o gerenciador do banco de dados e confirmou-se, nas três ocasiões, que a inserção ocorreu corretamente e no mesmo instante que foram enviados.

O teste (7) é semelhante ao anterior, e como resultado foram inseridos 244 novos registros no banco, e foi possível visualizar todos eles na página web. A foto 5 mostra o envio através da aplicação no Raspberry com a quantidade de registros inseridos retornada pelo servidor web, e os registros atualizados no banco de dados.

Foto 5 - Inserção no banco de dados

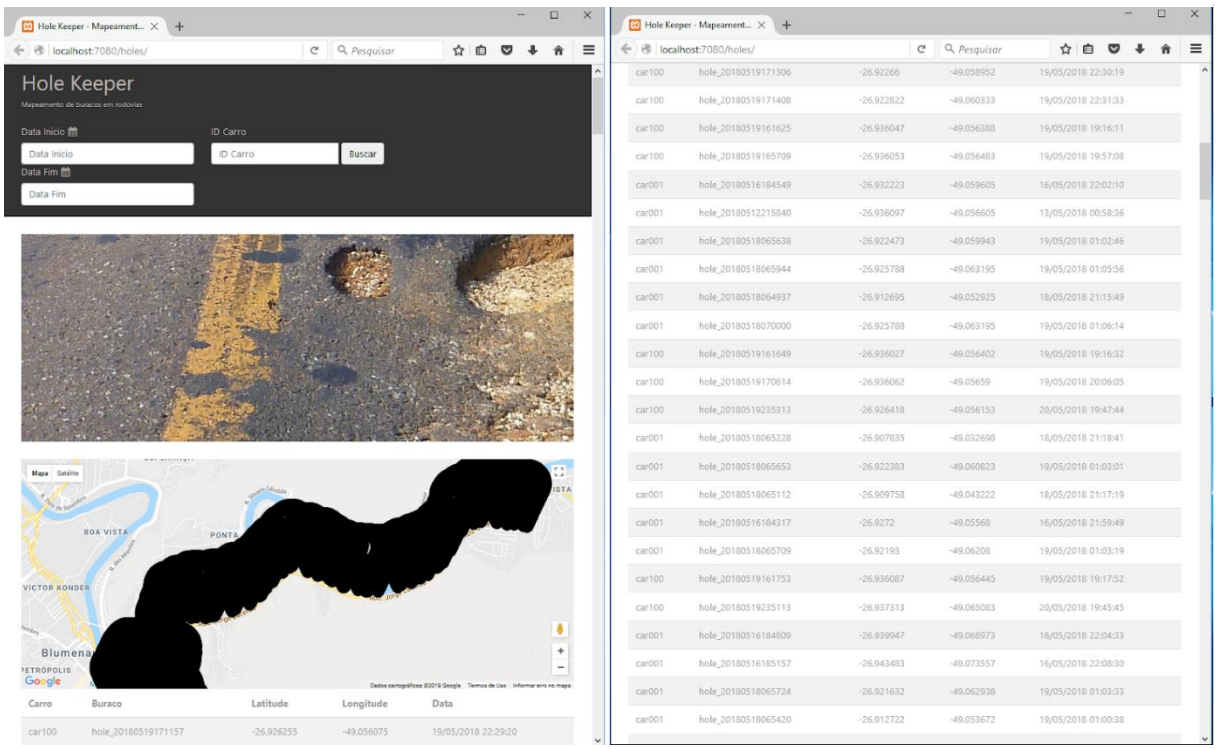


Fonte: Elaborado pelo autor (2018).

Com esse teste a persistência de dados foi colocada à prova. Nessa ocasião, o sistema demorou aproximadamente 1 minuto e 30 segundos para enviar todos os 244 registros armazenados, devido ao fato de ser inserido um registro por vez para garantir a atomicidade e a consistência dos dados, mas ao final, todos os dados foram inseridos.

Por fim, ao realizar o teste (8) do consumo desses dados através do *web service*, ao clicar para buscar todos os buracos na página, foram apresentados todos os registros encontrados tanto na tabela quanto no mapa, com os marcadores apresentando as informações individuais de cada buraco, e ao utilizar os filtros, foram apresentados somente os dados filtrados. A foto 6 mostra a página web carregada após enviar os dados para o servidor, com o mapeamento e a tabela de buracos. No início da visualização do mapa os marcadores ficaram bem próximos uns dos outros, mas à medida que se aproximava a imagem utilizando o zoom, foi possível identificar os buracos através de cada marcador (foto 7).

Foto 6 - Página web



Fonte: Elaborado pelo autor (2018).

Foto 7 - Mapeamento dos buracos

Fonte: Elaborado pelo autor (2018).

No teste final (9) foram encontrados 4 buracos. Com esse teste foi possível simular como o sistema se comportaria no ambiente real, desde a captura dos dados, a identificação do possível buraco até a apresentação dos dados ao usuário. O resultado foi positivo, pois foi possível visualizar os 4 buracos encontrados na página em tempo de execução, logo após o banco ter sido atualizado.

Os testes que foram realizados mostram o funcionamento da aplicação e demonstram a viabilidade do sistema para uso aplicado na detecção e mapeamento de buracos em rodovias, auxiliando na elaboração de programas para manutenção e melhorias de estradas.

4 CONSIDERAÇÕES FINAIS

O objetivo deste trabalho é gerar uma ferramenta capaz de captar as informações e a localização de possíveis buracos encontrados em rodovias, disponibilizando e demonstrando esses dados na forma de um mapeamento de buracos nas rodovias.

O sistema necessita passar por várias etapas para chegar até o mapeamento apresentado ao usuário, e o desenvolvimento de módulos e a integração entre eles foi essencial para que essa funcionalidade pudesse ser oferecida. Devido ao tamanho do projeto, não foi possível implementar todos os módulos necessários, não sendo desenvolvido apenas o algoritmo para identificação de buracos. No entanto, o objetivo foi alcançado.

Através dos módulos de Captação de dados e Servidor local embarcados em um carro, foi possível identificar potenciais buracos e armazenar localmente esses dados. Com a integração dos módulos Servidor local com o Servidor web, foi possível enviar esses dados e armazená-los na nuvem, onde o *web service* disponibiliza a consulta a essas informações, consumidas através de uma página web que apresenta um mapeamento dos buracos encontrados, buracos estes representados cada um por um marcador contendo as informações capturadas pelos sensores do módulo de Captação de dados.

Os testes realizados atestaram a atomicidade e a consistência dos dados do sistema, além da sua disponibilidade no ambiente testado. Entretanto, para garantir a atomicidade dos dados, o desempenho da aplicação em identificar os possíveis buracos foi diminuído, limitando em 10 segundos o intervalo necessário para encontrar um próximo buraco, fator que poderá ser aperfeiçoado após o desenvolvimento do algoritmo para identificar efetivamente um buraco.

Ao visualizar o mapeamento dos dados na página, é possível identificar os pontos no trajeto percorrido através da identificação do carro e do horário de captura dos dados, permitindo assim a utilização da ferramenta por múltiplos usuários.

Com essas informações é possível concluir que o sistema cumpre os objetivos delineados, captando as informações dos possíveis buracos e apresentando um mapeamento de forma que o usuário pode localizar em que parte do trajeto há um possível buraco, gerando valor de utilidade ao sistema desenvolvido.

Como trabalhos futuros, sugere-se o desenvolvimento de um algoritmo capaz

de identificar e classificar buracos usando os dados capturados pelo sensor acelerômetro; fazer uso de câmeras para auxiliar na identificação de buracos; a instalação dos módulos de captação nos quatro eixos de um carro, aumentando assim a performance do sistema em localizar buracos; e a implementação de uma solução para a inicialização do sistema e outro para enviar os dados para o servidor, de modo que o usuário não precise acessar o sistema operacional do Raspberry para efetuar o envio desses dados.

As melhorias sugeridas são para a página web, onde poderiam ser acrescentados outros tipos de filtros, otimizar a apresentação da tabela dos buracos, assim como implementar uma opção para buscar buracos por localidade e/ou região. Também seria interessante a implementação de um acesso à página como administrador, possibilitando a edição de dados, por exemplo a exclusão de buracos que foram recapeados ou registros muito próximos, que podem ser buracos duplicados.

Assim, o projeto desenvolvido pode ser utilizado como estrutura para um sistema completo de mapeamento de buracos em rodovias, no qual as melhorias realizadas pelos próximos trabalhos incrementariam a utilidade e usabilidade do sistema, fazendo com que a justificativa de se oferecer uma ferramenta útil tanto para quem necessita dar manutenção às rodovias quanto para quem é usuário sejam alcançadas.

REFERÊNCIAS

ALVES, Filipe Manuel Serra. **Micro inclinómetro eletromecânico**. 2012. 99 p. Dissertação (Mestrado) - Universidade do Minho, Escola de Engenharia, Mestrado em Engenharia Eletrónica Industrial e Computadores, Braga, Portugal, 2012. Disponível em: <http://intranet.dei.uminho.pt/gdmi/galeria/temas/pdf/52684.pdf>. Acesso em: 30 set. 2017.

ARDUINO. **Arduino.** 2017. Disponível em: < https://www.arduino.cc/>. Acesso em: 09 set. 2017.

BONJORNO, José Roberto *et al.* **Física**: mecânica, 1° ano. 2. ed. São Paulo: FTD, 2013.

BORGES, P. de S. *et al.* Embedded system for detecting and georeferencing holes in roads. **IEEE Latin America Transactions**, v. 9. n. 6. p. 921-925, 2011. Disponível em: < https://ieeexplore.ieee.org/document/6096973/>. Acesso em: 09 set. 2017.

CAVALCANTE, Marisa Almeida; TAVOLARO, Cristiane Rodrigues Caetano; MOLISANI, Elio. Física com Arduino para iniciantes. **Revista Brasileira de Ensino de Física**, v. 33, n. 4, p. 4503-4503, 2011. Disponível em: <http://www.sbfisica.org.br/rbef/pdf/334503.pdf >. Acesso em: 09 set. 2017.

CONFEDERAÇÃO NACIONAL DO TRANSPORTE. **Pesquisa CNT de rodovias 2016**: relatório gerencial. 20. ed. Brasília: CNT: SEST: SENAT, 2016. Disponível em: <http://pesquisarodoviascms.cnt.org.br/Relatorio%20Geral/Pesquisa%20CNT%20(2016)%20-%20LOW.pdf>. Acesso em: 05 ago. 2017.

DAL MORO, Tharcis; DORNELES, Carina; REBONATTO, Marcelo Trindade. Web services WS-* versus Web Services REST. **Revista de Iniciação Científica**, v. 11, n. 1, p. 39-51, 2009. Disponível em: <http://www.seer.ufrgs.br/reic/article/viewFile/22140/12928>. Acesso em: 09 set. 2017.

DUTRA, Andeise Cerqueira. **Análise comparativa de dados tomados a partir do sistema de posicionamento global (gps) utilizando a correção diferencial**. 2017. 75 p. Trabalho de conclusão de curso (Graduação) - Universidade Federal do Recôncavo da Bahia, Centro de Ciências Agrárias, Ambientais e Biológicas, Curso de Engenharia Florestal, Cruz das Almas, 2017. Disponível em: <http://www.repositoriodigital.ufrb.edu.br/bitstream/123456789/1150/1/TCC_GPS%20-%20ANDEISE%20final.pdf>. Acesso em: 09 set. 2017.

GARCIA, Cristiano M.; ABILIO, Ramon. Integração entre sistemas utilizando Web Services REST e SOAP: um relato prático. **Revista de Sistemas de Informação da FSMA**, n. 19, p. 34-41, 2017. Disponível em: <https://www.researchgate.net/profile/Cristiano_Garcia4/publication/320957382_Inte

gracao_entre_Sistemas_utilizando_Web_Services_REST_e_SOAP_Um_Relato_Pratico/links/5a0465074585151f479588bd/Integracao-entre-Sistemas-utilizando-Web-Services-REST-e-SOAP-Um-Relato-Pratico.pdf>. Acesso em: 30 set. 2017.

KWAPISZ, Jennifer R.; WEISS, Gary M.; MOORE, Samuel A. Activity recognition using cell phone accelerometers. **ACM SIGKDD Explorations Newsletter**. Volume 12 Issue 2, p. 74-82. New York: ACM. December 2010. Disponível em: <http://www.cis.fordham.edu/wisdm/public_files/sensorKDD-2010.pdf>. Acesso em: 09 set. 2017.

LEITE, Fabrício. **Construção de um inclinômetro para avaliar o efeito da declividade lateral no desempenho de tratores agrícolas**. 2007. 117 f. Tese (Doutorado) - Universidade Estadual Paulista, Faculdade de Ciências Agronômicas de Botucatu, Botucatu, 2007. Disponível em: <http://hdl.handle.net/11449/101950>. Acesso em: 30 set. 2017.

MOREIRA, Allyson dos Reis *et al.* **Sistema de detecção de buracos em estradas.** 2013. 44 p. Trabalho de conclusão de curso (Graduação) - Universidade Tecnológica Federal do Paraná, Departamento Acadêmico de Informática, Curso de engenharia de computação, Curitiba, 2013. Disponível em: <http://paginapessoal.utfpr.edu.br/msergio/portuguese/ensino-de-fisica/oficina-de-integracao-ii/oficina-de-integracao-ii/Relatorio%20Final%20-%20Buracos.pdf>. Acesso em: 05 ago. 2017.

MOROZ, Maiko R., JASINSKI, Ricardo P., PEDRONI, Volnei A. Requisitos para adoção de sistemas operacionais embarcados. **Visión Electrónica**, v. 6, n. 1, p. 90-97, jan. 2012. Disponível em: <https://dialnet.unirioja.es/descarga/articulo/4234855.pdf>. Acesso em: 30 set. 2017.

NASCIMENTO, Pedro Paulo Libório Lima do**. Desenvolvimento e avaliação de técnicas de localização para redes veiculares e sistemas de transportes inteligentes**. 2017. 73 p. Dissertação (Mestrado) - Universidade Estadual de Campinas, Instituto de Computação, Programa de Pós-Graduação em Ciência da Computação, Campinas, 2017. Disponível em: <http://repositorio.unicamp.br/jspui/handle/REPOSIP/325000>. Acesso em: 09 set. 2017.

OLIVEIRA, Maurício Pietrocola Pinto de. **Física**: conceitos e contextos: pessoal, social, histórico, movimento, força, astronomia. 1. ed. São Paulo: FTD, 2013. v. 1.

PANITZ, Mauri Adriano. **Dicionário técnico**: português-inglês. Porto Alegre: EDIPUCRS, 2003.

RAMALHO JUNIOR, Francisco; FERRARO, Nicolau Gilberto; SOARES, Paulo Antonio de Toledo. **Física 1**: os fundamentos da física: parte II. 10. ed. São Paulo: Moderna, 2009.

RAUTA, Leonardo *et al.* Proposta de sistemas embarcados para identificação de buracos em rodovias utilizando as Leis da Mecânica de Newton. In: COMPUTER ON THE BEACH, 8., 2017, Florianópolis, SC. **Anais...** Florianópolis, SC, 2017. p. 559-561. Disponível em: <https://siaiap32.univali.br/seer/index.php/acotb/article/view/10627/5965>. Acesso em: 28 out. 2017.

RICHARDSON, Matt; WALLACE, Shawn. **Primeiros passos com o Raspberry Pi**. 1. ed. São Paulo: Novatec, 2013.

ROCHA, Fábio Saraiva da; MARANGHELLO, Guilherme Frederico; LUCCHESE, Márcia Maria. Acelerômetro eletrônico e a placa Arduino para ensino de física em tempo real. **Caderno Brasileiro de Ensino de Física**, v. 31, n. 1, p. 98-123, 2013. Disponível em: <https://dialnet.unirioja.es/descarga/articulo/5165603.pdf>. Acesso em: 05 ago. 2017.

SILVA, Paulo Henrique da. **Desenvolvimento de um smart gateway para o monitoramento de máquinas industriais**. 2014. 85 p. Trabalho de conclusão de curso (Graduação) - Universidade do Vale do Itajaí, Centro de Ciências Tecnológicas da Terra e do Mar, Curso de Ciência da Computação, São José, 2014. Disponível em: <http://siaibib01.univali.br/pdf/Paulo%20Henrique%20da%20Silva.pdf>. Acesso em: 30 set. 2017.

SINDICATO NACIONAL DA INDÚSTRIA DE COMPONENTES PARA VEÍCULOS AUTOMOTORES; ASSOCIAÇÃO BRASILEIRA DA INDÚSTRIA DE AUTOPEÇAS. **Relatório da frota circulante 2017**. São Paulo: Sindicato Nacional da Indústria de Componentes para Veículos Automotores, 2017. Disponível em: <https://www.sindipecas.org.br/sindinews/Economia/2017/R_Frota_Circulante_2017.pdf>. Acesso em: 05 ago. 2017.

SOUZA, Anderson R. de *et al.* A placa Arduino: uma opção de baixo custo para experiências de física assistidas pelo PC. **Revista Brasileira de Ensino de Física**, v. 33, n. 1, p. 1702-1702, 2011. Disponível em: <http://sbfisica.org.br/rbef/pdf/331702.pdf>. Acesso em: 30 set. 2017.

SOUZA, Marco Antonio Furlan de; DENIS, Everson; FERNANDES, João Carlos Lopes. Utilização de um hardware embarcado (Raspberry PI) usando programação em blocos (Scratch) para ensino de física em escolas secundárias e universidades. In: CONGRESSO BRASILEIRO DE EDUCAÇÃO EM ENGENHARIA, 43., 2015, São Caetano do Sul. **Anais...** São Caetano do Sul, São Paulo, 2015. Disponivel em: <http://docplayer.com.br/34032492-Utilizacao-de-um-hardware-embarcado-raspberry-pi-usando-programacao-em-blocos-scratch-para-ensino-de-fisica-em-escolas-secundarias-e-universidades.html>. Acesso em: 05 ago. 2017.

TRINDADE, Derick Horrana da. **Monitoramento de sistemas de transporte com Arduíno e Shield-GSM, GPS, GPRS**. 2015. 31 p. Trabalho de conclusão de curso (Graduação) – Universidade de Brasília, Faculdade UnB Gama, Curso de

Engenharia Eletrônica, Brasília, 2015. Disponível em: <http://docplayer.com.br/8189566-Universidade-debrasilia-unb-faculdade-unb-gama-fga-curso-de-engenharia-engenhariaeletronica.html/>. Acesso em: 05 ago. 2017.