Uma arquitetura de *Beacons* customizáveis para Internet das Coisas

Emmanuel C. A. de Melo[1], Marcio E. F. Maia[1], Paulo A. L. Rego[1], Francisco H. C. S. Filho[1], Rossana M. C. Andrade[2]

[1]Universidade Federal do Ceará (UFC)
Campus Quixadá
Av. José de Freitas Queiroz, 5003 – Cedro Novo, Quixadá – CE, 63900-000 – Brasil

[2]Universidade Federal do Ceará (UFC)
Departamento de Computação
Campus do Pici - Centro de Ciências - Bloco 910, Fortaleza – CE, 60440-900 – Brasil

{cainammello, marcioefmaia, pauloalr, helderhdw, rossana}@ufc.br

Abstract. *The Internet of Things (IoT) can be defined as the set of technologies connecting objects from the physical world, empowering them with the ability to react to events produced by human and other systems. In that direction, the goal of this paper is to present an architecture to build Customizable Beacons, allowing user devices and Smart Objects to directly access the data published by the Beacons, without requiring Internet connection. The main contributions of this paper are the creation of an customizable Beacon using Arduino and BLE; an Android API to interact with Beacons and a server to manage the information published by each Beacon.*

Resumo. *A Internet das Coisas (Internet of Things - IoT) pode ser definida como um conjunto de tecnologias que conecta objetos do mundo físico com a Internet, tornando-os capazes de reagir a eventos produzidos por humanos ou outros sistemas. Assim, o objetivo desse artigo é apresentar uma arquitetura para a construção de Beacons customizáveis, permitindo que dispositivos de usuários e outros Objetos Inteligentes (OIs) possam acessar diretamente os dados publicados pelo Beacon, sem a necessidade de conexão com a Internet. As principais contribuições desse artigo são a criação do Beacon customizável utilizando o Arduino e BLE; uma API Android para interagir com os Beacons e um servidor para atualizar os dados publicados por cada Beacon.*

1. Introdução

A Internet das Coisas (*Internet of Things* - IoT) pode ser definida como um conjunto de tecnologias que conecta objetos do mundo físico com a Internet, tornando-os capazes de reagir a eventos produzidos por humanos ou outros sistemas [Evangelatos et al. 2012]. Nesse sentido, objetos inteligentes são objetos do mundo físico embarcados com algum tipo de processamento e comunicação. Dentro da IoT, prédios inteligentes utilizam os Objetos Inteligentes (OIs) para promoverem serviços que aumentam o conforto de seus ocupantes e a eficiência das tarefas realizadas [Agarwal and Weng 2012]. Exemplos de serviços oferecidos nos prédios inteligentes podem ser o controle de acesso às dependências do prédio, informações sobre localização e navegação *indoor*, conteúdo das salas e propagandas customizadas.

Nesse sentido, muitos desses serviços oferecidos pelos prédios inteligentes são baseados na interação direta entre usuários e OIs. Essa interação direta normalmente ocorre através de tecnologias de comunicação baseada em proximidade (e.g., Bluetooth, Bluetooth Low Energy, Zigbee e NFC) e permite que os OIs divulguem sua identidade ou os dados coletados, e ainda atuem no ambiente. Assim, para facilitar a criação de serviços que utilizam a interação direta entre OIs e usuários, o conceito de *Beacon* foi criado, permitindo que uma determinada informação seja publicada periodicamente e capturada por dispositivos de usuário (*smartphones*) e de ambiente (outros OIs), com um mínimo de consumo de energia. Um problema importante nas implementações atuais de *Beacons* é a dificuldade de customizar o conteúdo publicado por cada *Beacon*. No geral, os *Beacons* publicam seus identificadores estáticos, e a informação que esses identificadores representam é acessada na Internet.

Entretanto, em ambientes onde muitos dispositivos acessam informações sobre *Beacons*, o acesso à Internet para buscar as informações representadas pelo *Beacon* pode gerar problemas de desempenho, principalmente se a quantidade de *Beacons* for grande. Ou ainda impossibilitar o uso de *Beacons* em aplicações onde a conexão com a Internet é intermitente. Nesse contexto, o objetivo desse artigo é apresentar uma arquitetura para a construção de *Beacons* customizáveis, onde uma parte dos dados coletados pelos OIs são publicados periodicamente juntamente com o identificador do dispositivo. Essa abordagem permite que dispositivos de usuário e outros OIs acessem diretamente os dados publicados pelo *Beacon* sem a necessidade de conexão com a Internet, permitindo a criação de serviços em uma escala micro-local ou serviços de localização *indoor*.

A arquitetura apresentada no artigo foi implementada utilizando um hardware de prototipação com uma interface de comunicação Bluetooth Low Energy (BLE) para a criação do *Beacon* e publicação dos dados para usuários e outros OIs próximos, uma interface com acesso à Internet para a atualização, quando necessário, dos conteúdos publicados pelo *Beacons*, um conjunto de interfaces de acesso para dispositivos Android na forma de uma *Application Programming Interface* (API) para facilitar a detecção e o acesso por uma aplicação aos dados publicados por um *Beacon*, além de uma interface web para que usuários possam atualizar o conteúdo dos *Beacons*. A arquitetura implementada foi validada através de um estudo de caso e avaliação de desempenho.

Este trabalho está organizado da seguinte forma: a Seção **2** apresenta os principais conceitos utilizados por esse trabalho. A Seção **3** descreve os trabalhos relacionados. Na Seção **4**, é discutida a arquitetura proposta e sua implementação. A Seção **5** apresenta o estudo de caso criado e os resultados da análise de desempenho. Por fim, a Seção **6** traz as considerações finais e os trabalhos futuros.

2. Referencial Teórico

Nesta seção são apresentados alguns conceitos básicos importantes para que se possa compreender o contexto no qual esta proposta está inserida, e o motivo pelo qual certas decisões foram tomadas, como escolha de equipamentos, tecnologias e protocolo de comunicação.

2.1. Bluetooth de Baixa Potência

Bluetooth de Baixa Potência (*Bluetooth Low Energy* - BLE) ou *Smart Bluetooth* é um tipo de tecnologia de conexão sem fio desenvolvida pelo Grupo de Interesse Especial

sobre *Bluetooth* (*Bluetooth Special Interest Group*) e pode ser entendida como uma especificação, ou funcionalidade que integra o *Bluetooth* desde 2010, a partir de sua versão 4.0 [Bluetooth 2010b]. O BLE possui algumas vantagens comparado ao *Bluetooth* Clássico, que o torna mais "inteligente" ao proporcionar mais funcionalidades para os desenvolvedores, fabricantes e para os usuários que terão uma melhor experiência com IoT. Estas vantagens fazem do BLE uma das tecnologias mais recomendadas para se empregar a Internet das Coisas [Bluetooth 2010a], como as listadas a seguir:

- Baixo consumo de energia, podendo funcionar por até quase um ano com uma célula de bateria do tamanho de uma moeda;
- Capacidade de múltiplas conexões, onde um dispositivo BLE central, chamado de BLE *master*, é responsável por aceitar, gerir e controlar as conexões de vários outros dispositivos que irão requisitar conexão, e são conhecidos como BLE *slaves*;
- Capacidade de enviar pequenas quantidade de bytes em *broadcast* para diversos dispositivos ao mesmo tempo sem a necessidade de criar uma conexão;
- Fornecer informações de sensores dos dispositivos que o utilizam, como: porcentagem da bateria, heixo de rotação ou RSSI (*Received Signal Strength Indicator*), através do serviço de *response* sobre a mensagem em *broadcast* enviada;
- Baixo custo, encontrando no mercado, produtos com a tecnologia por menos de $10,00 dólares.

O BLE vem sendo muito utilizado em sistemas de localização *indoor*. A *Apple* utilizou a tecnologia do BLE para lançar o *Ibeacon*, um dispositivo capaz de localizar dispositivos IOS (*Iphone Operating System*) próximos que possuam a tecnologia BLE e enviar-lhes informações com base na localização desses dispositivos [Conte et al. 2014]. Para o futuro, é previsto que o BLE esteja presente em mais de 1 bilhão de dispositivos devido ao crescente uso da tecnologia do *Bluetooth* Clássico, utilizada em celulares, carros, notebooks, e diversos outros dispositivos. Esta popularização do *Bluetooth* Clássico serve como uma espécie de alavanca para impulsionar o crescimento da tecnologia do BLE [Gomez et al. 2012].

2.2. Beacons BLE

Um *Beacon* BLE pode ser entendido como um dispositivo que utiliza da Tecnologia do *Bluetooth Low Energy* para transmitir uma mensagem em *broadcast* para todos os dispositivos que estejam em um raio de cerca de 100 metros e possuam a função BLE ativada [Zafari and Papapanagiotou 2015]. Os dispositivos, geralmente *smartphones* e *tablets*, reagem a essa mensagem realizando alguma ação referente à mensagem recebida. É importante frisar que *Beacons* são dispositivos periféricos de comunicação e não devem ser utilizados para trabalhar com grandes volumes de dados. Como disse Patrick Leddy, chefe executivo e fundador da empresa *Pulsate*, os *Beacons* funcionam apenas como uma espécie de farol (por isso a nomenclatura "Beacon", do português, "faról"), apenas enviando mensagens com uma pequena quantidade de dados em *broadcast* para os dispositivos ao redor, os informando sobre sua existência [Statler 2016].

A Figura 1 apresenta como uma mensagem enviada por um *Beacon BLE* é estruturada, com seu tamanho podendo variar de 8 a 47 bytes [Bluetooth 2010a]. O conceito de *Beacon* ficou mais conhecido com o aparecimento do *IBeacon*, criado pela *Apple*. O *IBeacon* é um protocolo específico baseado na tecnologia do BLE e se

destina e ser usado como um *BLE Beacon* voltados para os dispositivos da empresa [Zafari and Papapanagiotou 2015].

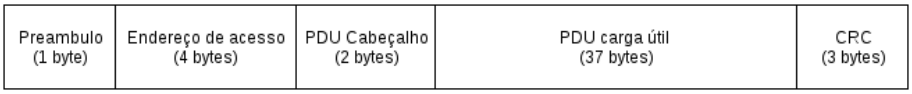


Figura 1. Estrutura do pacote de uma mensagem BLE

A Figura 2 apresenta como o protocolo IBeacon está estruturado dentro do pacote de mensagem BLE. Dentro do PDU carga útil (Figura 2.2), o endereço MAC (*Media Access Control*) contém o identificador único do dispositivo que está mandando a mensagem e 31 bytes restantes para serem utilizados. Na Figura 2.3, a estrutura do pacote *Ibeacon* é apresentada, que ocupa 30 bytes dos 31 disponíveis.

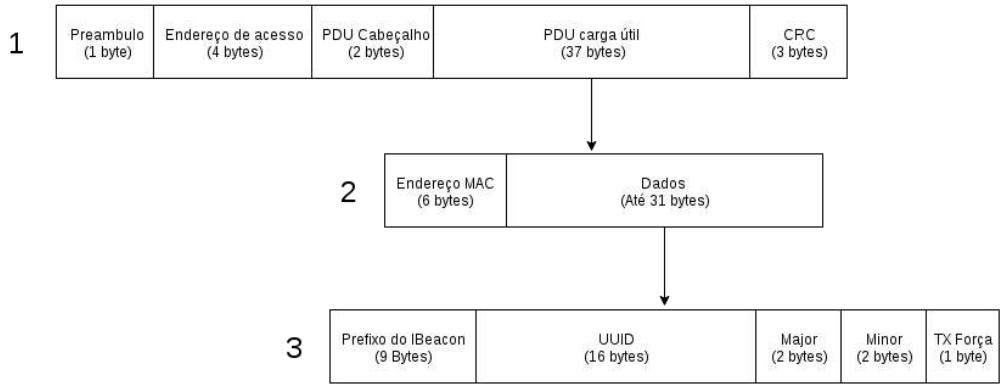


Figura 2. Protocolo IBeacon dentro da estrutura do pacote da mensagem BLE

2.3. Message Queue Telemetry Transport

Message Queue Telemetry Transport (MQTT) é um protocolo de comunicação extremamente leve e rápido, desenvolvido originalmente pela empresa IBM e é utilizado principalmente na comunicação M2M (*machine-to-machine*) [Lampkin et al. 2012]. Além de ser extremamente rápido e simples, o MQTT apresenta diversos benefícios para dispositivos com limitações de recursos (dispositivos embarcados) [MQTT 2016], como:

- Troca de mensagem assíncrona, sem a dependência de resposta ou mensagens de confirmação;
- Tamanho reduzido das mensagens trocadas, já que cabeçalho do pacote enviado com o protocolo MQTT é extremamente simples, deixando o pacote basicamente com os dados desejados, o que é essencial em casos de largura de banda baixa;
- Baixo uso dos recursos de processamento e memória;
- Comunicação otimizada para sensores e dispositivos remotos;
- Alta escalabilidade, através da utilização do esquema de tópicos e do padrão *publish/subscriber* (publicadores/assinantes).

O protocolo MQTT implementa a arquitetura *publish/subscriber* na sua comunicação, com um servidor central conhecido como *broker* (intermediário). O protocolo gira em torno do conceito de tópicos, onde todos os clientes se inscrevem, tanto para enviarem

informações, como para recebê-las [Lampkin et al. 2012]. De acordo com a arquitetura *publish/subscriber*, uma comunicação através do protocolo MQTT funciona da seguinte maneira: o *publisher* publica as informações para o *broker* (servidor) através de um tópico; os *subscribers* inscritos nesse mesmo tópico recebem essas informações do *broker*.

Tanto *publishers* como *subscribers* são clientes na arquitetura implementada pelo MQTT, e conectam-se a apenas um *broker*. Porém, um mesmo cliente MQTT pode ser *subscriber* e *publisher* de diversos tópicos [Lampkin et al. 2012]. Uma mensagem no protocolo MQTT é dividida em duas partes: **tópico** e ***payload***. Um tópico é uma *string* (similar a uma *url*) e é utilizada para implantar uma hierarquia entre os tópicos, o que facilita a capacidade de um cliente publicar e se inscrever em um tópico especifico, ou mesmo em todos os tópicos de uma subárea, um exemplo de tópico: area/area_id/sensor/sensor_id/. Por fim, temos a ultima parte do pacote, o *payload*, onde se encontra a informação propriamente dita [Hunkeler et al. 2008].

3. Trabalhos Relacionados

Em Khoo [Khoo 2015], uma solução para automação de conexões móveis é proposta para interação com um dispositivo de vídeo através de um servidor, que por sua vez deve gerenciar estas conexões. O autor escolhe a ferramenta *IBeacon* da *Apple*, equipada com a tecnologia BLE, assim como a tecnologia BPAN (*Bluetooth Personal Area Network*), uma outra especificação do *Bluetooth*. O autor utiliza a tecnologia BPAN para prover acesso a um servidor através de uma rede de conexões IP (*Internet Protocol*), sobre o *Bluetooth*. O autor também estrutura a plataforma criada no modelo cliente-servidor, onde uma placa *Raspberry Pi* faz a função de servidor, enquanto uma aplicação desenvolvida para *Android* faz o papel de cliente.

Em Conte *et al* [Conte et al. 2014], os autores propõem uma solução para detecção de ocupantes de uma construção utilizando a plataforma chamada *BLUE-SENTINEL* para localizar e quantificar os ocupantes na construção e utilizar estas informações para aprimorar as decisões tomadas pelo construção inteligente. A plataforma criada consiste na implementação do protocolo *IBeacon* em um *Beacon* customizado, utilizando uma placa Arduino. O *Beacon* criado é então utilizado para mapear os locais do prédio e enviar a um servidor denominado *BLUE-SENTINEL Core* informações sobre a distância do aparelho e sobre o usuário. O servidor irá repassar essas informações para o Sistema Gerenciador da Construção (SGC) que irá tomar a melhor decisão com base nessas informações.

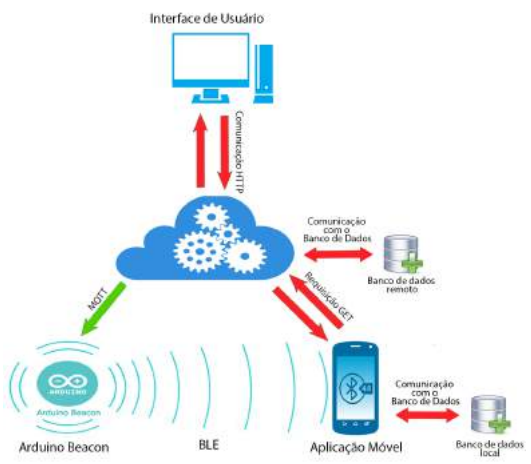
Em Takalo *et al* [Takalo-Mattila et al. 2013] é apresentada uma abordagem para representar semanticamente objetos físicos reais utilizando *Smart Bluetooth Beacons* para mapear esses objetos e enviar um identificador único chamado de *ucode* para a aplicação do usuário. A aplicação por sua vez, requer a resolução do *ucode* por um servidor e após recuperar esta informação, o servidor informa o endereço no banco de dados referente ao objeto de interesse. Por fim, a aplicação do usuário utiliza este endereço para consultar o banco de dados e obter as informações desejadas sobre o determinado objeto.

O principal problema dos trabalhos apresentados é a dependência a um servidor para que o sistema possa ser executado. De forma contrária, o trabalho aqui proposto já publica informações sobre os dados coletados pelo *Beacon* diretamente para os dispositivos próximos, diminuindo a dependência com servidores externos e aumentando a quantidade de interações locais.

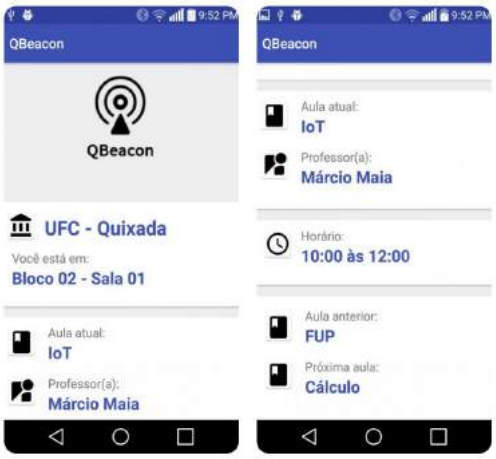
4. Arquitetura do Sistema

Esse artigo propõe uma arquitetura para a criação de *Beacons* customizáveis para IoT. O objetivo é permitir que a informação publicada pelos *Beacons* possa ser atualizada em tempo de execução, e ainda facilitar o desenvolvimento de aplicações Android que acessam essas informações. A arquitetura proposta consiste em três módulos, como mostrado na Figura 3(a). Todos os códigos utilizados nesse projeto podem ser encontrados em *https://github.com/cainammello*.

O módulo Aplicação Móvel é responsável por mostrar ao usuário as informações enviadas pelo *Beacon*. A Figura 3(b) mostra a Aplicação *Android* acessando tais informações. A aplicação percorre a mensagem enviada pelo *Beacon* e extrai as informações contidas nele. No Algoritmo **1**, vemos uma parte de como isso ocorre. No primeiro método, os *Beacons* próximos ao dispositivo são reconhecidos e inicializados. No segundo método, os dados que vêm do *Beacon* são utilizados para carregar os objetos que serão usados para mostrar ao usuário as informações sobre as salas de aula.



(a) Arquitetura do sistema proposto baseada em cliente-servidor



(b) Campos atualizados após o recebimento da mensagem pelo Beacon

Figura 3. Arquitetura da proposta e Tela principal do módulo Aplicação Android

```
beaconManager.addRangeNotifier(new RangeNotifier() {
    @Override
    public void didRangeBeaconsInRegion(Collection<Beacon> beacons, Region
        region) {
        if (beacons.size() > 0) {
            Beacon b = beacons.iterator().next();
            while (b != null) {
                onBeaconReceived(b);
                b = beacons.iterator().next();
            }
        }
    }
});

public void onBeaconFinded(Beacon b, String uuid) {
    QBeaconProtocolo beaconProtocolo = new QBeaconProtocolo();

    Sala sala = beaconProtocolo.getObjectFrom(Sala.class,uuid,1);
```

```
18      Bloco bloco = beaconProtocolo.getObjectFrom(Bloco.class,uuid,1);
19      Docente docente = beaconProtocolo.getObjectFrom(Docente.class,uuid,1);
20      Disciplina disciplina = beaconProtocolo.getObjectFrom(Disciplina.class,
            uuid,1);
21      Instituicao instituicao = beaconProtocolo.getObjectFrom(Instituicao.
            class,uuid,1);
22      Campus campus = beaconProtocolo.getObjectFrom(Campus.class,uuid,1);
23      Disciplina aulaAnt = beaconProtocolo.getObjectFrom(Disciplina.class,uuid
            ,2);
24      Disciplina aulaProx = beaconProtocolo.getObjectFrom(Disciplina.class,
            uuid,3);
25      Integer hora = beaconProtocolo.getValueFrom(uuid,1);
26  }
```

Algoritimo 1. Recebendo e utilizando os dados do *Beacon* para inicializar os objetos que serão utilizados para mostrar as informações ao usuário.

O módulo *Web Service* foi desenvolvido com base na arquitetura REST (*REpresentational State Transfer*) utilizando a linguagem de programação *JavaScript*, através do *framework* NodeJS[1], e é responsável por cadastrar, armazenar e atualizar as informações adicionadas sobre os *Beacons*. Estas informações são "chaves" contidas no identificador do *Beacon*, e são salvas no banco de dados local do dispositivo móvel para serem acessadas quando a conexão estiver indisponível e são atualizadas quando a conexão voltar a estar disponível.

O módulo Arduino *Beacon* é responsável por publicar os dados, usando a interface BLE, para a Aplicação Móvel, que consome esses dados. Um dos problemas encontradas nos modelos pesquisados é a dependência da conexão com o servidor web que a aplicação *mobile* possui, para obter os dados, impossibilitando o uso quando a conexão não está disponível. Entretanto, neste trabalho, os dados são transmitidos diretamente pelo *Beacon* e as informações que são grandes demais para serem publicados por ele, são acessadas diretamente via banco de dados local, que é atualizado quando o aplicativo é utilizado pela primeira vez, e posteriormente quando as informações forem atualizadas no servidor. É importante mencionar que o acesso ao servidor pelo aplicativo acontece com uma frequência muito baixa.

O módulo *Arduino Beacon* é formado por um *Bluetooth* de Baixa Potência, de referência HM-10, um *Shield Ethernet*, de referência HanRun HR911105A e a própria placa de prototipagem Arduino Uno, como mostra a Figura 4. O módulo é encarregado de transmitir as informações capturadas por sensores Arduino através do BLE.

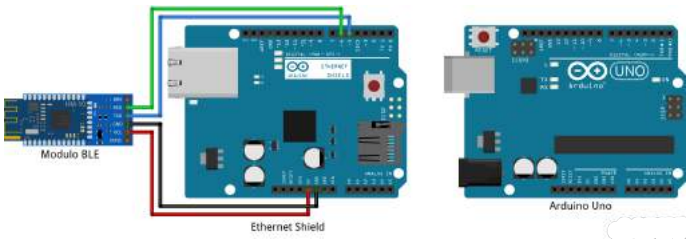


Figura 4. Circuito módulo BLE mais Ethernet Shield e Arduino UNO

A comunicação entre *Arduino Beacon* e *Web Service* ocorre através de um tópico

[1]NodeJS: https://nodejs.org/en/.

MQTT ao qual um *Beacon* está subscrito, como explicado na seção 2.3. Esses tópicos são criados pelo *Web Service* em tempo de execução, de acordo com as informações cadastradas sobre um *Beacon*. Para passar transmitir uma nova informação, o *Beacon*, após se inscrever no tópico MQTT correto, recebe do Broker (*Web Service*) as mensagens que são publicadas naquele tópico e as envia em modo *broadcast* pela interface BLE. A aplicação *mobile* por sua vez, recebe a mensagem enviada em *broadcast* através da biblioteca AltBeacon[2] e as mostra ao usuário.

O protocolo *IBeacon* foi utilizado com o objetivo de aproveitar os bytes que eram utilizados apenas como identificador, e substituí-los por identificadores (IDs), que representam os dados armazenados no banco de dados local da Aplicação Móvel. Cada ID tem um tamanho de 2 hexadecimais para cada dado publicado. A Tabela 1 nos mostra como é estruturado o UUID e os comandos **AT** (comandos Hayes) necessários para modifica-los. Os comandos **AT** são passados através de uma comunicação serial entre a interface do *Bluetooth Low Energy* e o Módulo BLE.

Tabela 1. BLE Universally Unique Identifier (UUID)

UUID: 00000001-00000001-0000-0001-00000001			
AT+IBE000000001	AT+IBE100000001	AT+IBE200000001	AT+IBE300000001

Com a mensagem enviada pelo *Web Server*, o Arduino *Beacon* quebra essa mensagem em 4 conjuntos de 8 hexadecimais (ou 4 bytes) e armazena cada um em uma faixa do UUID do BLE, seguindo o padrão formado na hora do cadastro das informações no *Web Service*. Depois de ter dividido as informações, o mapeamento dos dados publicados pelo *Beacon* mostrado na Tabela 2 é realizado.

Tabela 2. Adaptação do protocolo IBeacon UUID

UUID				
	ID de sala	ID de bloco	ID de disciplina	Hora de início
AT+IBE0:	00	00	00	00
	ID de docente	ID de docente	ID de instituição	Minutos de início
AT+IBE1:	00	00	00	00
	ID de campus	ID de aula anterior	ID de próxima aula	Hora de fim
AT+IBE2:	00	00	00	00
	ID de histórico	ID livre	ID livre	Minuto de fim
AT+IBE2:	00	00	00	00

5. Avaliação da Proposta

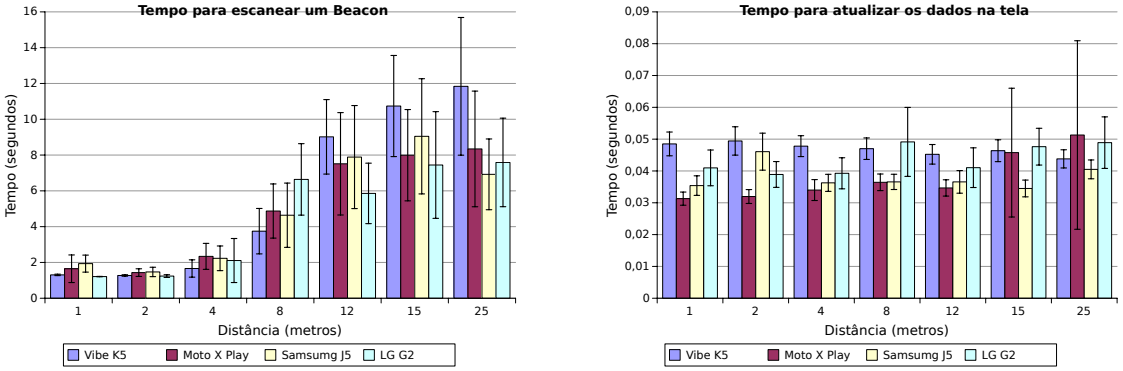
Com o objetivo de analisar como a proposta se comporta com dispositivos reais, duas avaliações foram realizadas: i) um estudo de caso que fornece um guia das aulas e salas em um campus inteligente; e ii) uma avaliação de desempenho que mede o tempo de detecção de novos *Beacons* e o tempo de leitura dos dados publicados por eles. O estudo de caso implementado fornece informações sobre salas de aula, professores e conteúdo das aulas em um campus inteligente. Nele, uma aplicação Android acessa as informações publicadas por um *Beacon*, a fim de descrever as informações referentes à sala que o usuário está inserido. A Figura 3(b) e o Algoritmo **1** apresentam uma parte desse estudo de caso. O código completo está disponível em *https://github.com/cainammello*.

[2]AltBeacon: http://altbeacon.org/.

5.1. Avaliação de Desempenho

A avaliação de desempenho foi realizada com quatro dispositivos móveis: Lenovo Vibe K5, Samsumg J5, LG G2 e Motorola Moto X Play. Os experimentos foram repetidos 30 vezes para cada combinação de dispositivo móvel e distância (que variou de 1 a 25 metros). A Figura 5 (a) apresenta a média do tempo de detecção do *Beacon*, com 95% de confiança, para diferentes distâncias entre os dispositivos e o *Beacon*. Pode-se perceber que, para todos os dispositivos móveis, quanto maior a distância entre o *smartphone* e o *Beacon*, maior o tempo de detecção. No melhor caso (distância de 1m) o *Beacon* foi detectado em menos de 2s, enquanto no pior cenário (distância de 25m) o *Beacon* levou em média 12s para ser descoberto (no Vibe K5).

A Figura 5 (b) apresenta a média do tempo que os dispositivos móveis levam para atualizar a tela uma vez que o *Beacon* é descoberto, com 95% de confiança. Pode-se perceber que, ao contrário do tempo para detectar o *Beacon*, a variação da distância não impacta no tempo de atualização da tela. Uma vez que o *Beacon* é detectado, as informações são extraídas do *Beacon* e do banco de dados local, o que requer um tempo aproximadamente constante. O experimento mostrou também que tal processo é rápido, demorando menos de um décimo de segundo em todos os dispositivos móveis.



(a) Tempo para reconhecer *Beacon* x distância

(b) Tempo para atualizar os dados x distância

Figura 5. Resultados da avaliação de desempenho

6. Considerações Finais e Trabalhos Futuros

Esse artigo apresentou uma uma arquitetura para a criação de *Beacons* customizáveis para Internet das Coisas. O objetivo era permitir que a informação publicada pelos *Beacons* pudesse ser acessada de forma direta por proximidade, sem a necessidade de conexões com servidores na Internet. Adicionalmente, o desenvolvimento de aplicações Android que acessam a essas informações foi facilitado, fornecendo uma API para acessar as informações dos *Beacons*.

As principais contribuições desse artigo são a criação do *Beacon* customizável utilizando o Arduino e BLE; uma API Android para facilitar o desenvolvimento de aplicações que acessam *Beacons*; um servidor, com um cliente *front-end*, para atualizar os dados publicados pelos *Beacons*. Essas contribuições foram validadas de duas formas diferentes: i) um estudo de caso que fornece um guia das aulas e salas em um campus inteligente; e ii) uma avaliação de desempenho que mede o tempo de detecção de novos *Beacons* e o tempo de leitura dos dados publicados por eles,

Como trabalhos futuros, as seguintes melhorias serão realizadas: propor um meio de tornar mais genérico a forma com que o mapeamento dos bytes no UUID são feitos; otimizar o *response* do BLE para conseguir mandar mais dados sem deixar o processo de comunicação lento; e colocar a plataforma em execução com o objetivo de coletar dados sobre o seu funcionamento.

Referências

Agarwal, Y. and Weng, T. (2012). From buildings to smart buildings—sensing and actuation to improve energy efficiency. *IEEE Design & Test of Computers*, vol.29(4):36–44.

Bluetooth, S. (2010a). The bluetooth core specification, v4. 0. *Bluetooth SIG: San Jose, CA, USA*.

Bluetooth, S. (2010b). Specification of the bluetooth system-covered core package version: 4.0.

Conte, G., De Marchi, M., Nacci, A. A., Rana, V., and Sciuto, D. (2014). Bluesentinel: a first approach using ibeacon for an energy efficient occupancy detection system. In *BuildSys@ SenSys*, pages 11–19, 1st ACM International Conference on Embedded Systems For Energy-Efficient Buildings (BuildSys) 2014. ResearchGate.

Evangelatos, O., Samarasinghe, K., and Rolim, J. (2012). Evaluating design approaches for smart building systems. In *Mobile Adhoc and Sensor Systems (MASS), 2012 IEEE 9th International Conference on*, pages 1–7, Las Vegas, NV. IEEE, IEEE.

Gomez, C., Oller, J., and Paradells, J. (2012). Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753.

Hunkeler, U., Truong, H. L., and Stanford-Clark, A. (2008). Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In *Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on*, pages 791–798. IEEE.

Khoo, K. C. Y. (2015). *Automation of internet of things connection*. PhD thesis, UTAR.

Lampkin, V., Leong, W. T., Olivera, L., Rawat, S., Subrahmanyam, N., Xiang, R., Kallas, G., Krishna, N., Fassmann, S., Keen, M., et al. (2012). *Building smarter planet solutions with mqtt and ibm websphere mq telemetry*. IBM Redbooks.

MQTT, O. (2016). Mqtt: Mq telemetry transport.

Statler, S. (2016). Geofencing: Everything you need to know. In *Beacon Technologies*, pages 307–316. Springer.

Takalo-Mattila, J., Kiljander, J., and Soininen, J.-P. (2013). Advertising semantically described physical items with bluetooth low energy beacons. In *Embedded Computing (MECO), 2013 2nd Mediterranean Conference on*, pages 211–214, 2013 2nd Mediterranean Conference on Embedded Computing (MECO), Budva. IEEE, IEEE.

Zafari, F. and Papapanagiotou, I. (2015). Enhancing ibeacon based micro-location with particle filtering. In *Global Communications Conference (GLOBECOM), 2015 IEEE*, pages 1–7. IEEE.