EAComp
Encontro Acadêmico de Computação

VI Encontro Acadêmico de Computação - UFMA
Copyright (c) EAComp. ISSN: 2594-486X

Aplicação da arquitetura SOA em sistemas embarcados para IoT

João Higo Sousa Nunes, João Vitor Miranda Roma, Luis Claudio de Oliveira Silva, Inaldo Capistrano Costa

[1]Coordenação de Engenharia da Computação – Universidade Federal do Maranhão (UFMA)
Av. dos Portugueses, 1966 Bacanga – 65.080-805 – Sao Luís – MA – Brasil

higo.sousaa@gmail.com, jvmr2.0@gmail.com,
lcoliveira.silva@gmail.com, inaldo.costa@gmail.com

Abstract. *The Internet of Things (IoT) is a technological paradigm that aims to connect electronic devices to each other and to the Internet. For this connection, it is emphasized the use of sensors and actuators in conjunction with embedded systems microcontrollers connected in a network. This work proposes to build a low cost system for the Internet of Things through the Service Oriented Architecture. In order to achieve this goal, a prototype was built to serve as a proof of concept for the use of REST as an integrating technology among several platforms, demonstrating its capacity to expand to several applications.*

Resumo. *A Internet das Coisas (em inglês, IoT) é um paradigma tecnológico que tem como objetivo conectar dispositivos eletrônicos entre si e com a Internet. Para essa conexão, é ressaltado o uso de sensores e atuadores em conjunto com sistemas embarcados microcontroladores conectados em uma rede. Este trabalho se propõe a construir um sistema de baixo custo para IoT por meio da Arquitetura Orientada a Serviços. Para alcançar esse objetivo, neste trabalho foi construído um protótipo que serve como prova de conceito para o uso do REST como tecnologia integradora entre várias plataformas, demonstrando sua capacidade de expansão para diversas aplicações.*

1. Introdução

Facilitar a execução de suas tarefas cotidianas sempre foi uma busca do ser humano. A Internet das Coisas (em inglês, *Internet of Things*, ou IoT) é um paradigma tecnológico que tem como objetivo conectar dispositivos eletrônicos entre si e com a Internet, permitindo a coleta e a troca de dados entre eles, desta forma, criando maior integração entre os sistemas e o mundo real, em busca de melhorias na eficiência dos serviços oferecidos aos humanos [Atzori et al. 2010].

Este trabalho se propõe a demonstrar a viabilidade e vantagens de aplicação da arquitetura SOA no ambiente IoT. Para isso, foi elaborada uma arquitetura que visa integrar *Web Services* com sistemas embarcados e dispositivos móveis. A solução proposta precisa ser suportada por dispositivos acessíveis em termos de custo. Adicionalmente, não deve requerer muitos recursos de hardware, pois pretende-se que o sistema seja de baixo custo, o que implica em pouco poder de processamento e uso de padrões não proprietários. Dessa forma, após a definição das tecnologias a serem adotadas, foi desenvolvida uma prova de conceitos na qual foi implementada uma aplicação que demostrou a viabilidade da arquitetura proposta.

2. Fundamentação Teórica

Nesta seção são definidos os principais conceitos associados à solução proposta.

2.1. Internet das Coisas

> ”Se tivéssemos computadores que soubessem tudo o que havia para saber sobre as coisas - utilizando dados que elas mesmo reuniriam de forma independente - seríamos capazes de monitorar e contar tudo, e reduzir muito o desperdício, perdas e custos. Nós saberíamos quando as coisas precisariam de substituição, reparo ou troca, e se elas estariam novas ou se já passaram da sua vida útil.” [Ashton 2009]

A IoT pode ser aplicada a casas, prédios, indústrias, transportes, hospitais, sistemas de gerenciamento de água, energia e alimentos, entre outros ambientes, tornando-os inteligentes, ou seja, utilizando a computação de forma imperceptível para melhorar suas atividades. Para o desenvolvimento de um sistema IoT, é ressaltado o uso de sensores e atuadores conectados em uma rede, colentando e enviando informações. Isso pode ser usado para monitorar e avaliar características de um ambiente, como a temperatura e os horários com maior ou menor frequência de movimento no local [Friess 2013].

2.2. Arquitetura Orientada a Serviços

O paradigma SOA consiste em um software produtor e um consumidor de serviços. O consumidor manda uma mensagem com a requisição do serviço ao produtor, este, por sua vez, responde ao consumidor com uma mensagem contendo o serviço. As mensagens são definidas de uma forma que sejam compreensíveis tanto para o consumidor quanto para o produtor de serviços. Um produtor também pode ser um consumidor de um outro serviço [Ferreira 2015].

Ao se falar em troca de serviços entre plataformas heterogêneas, é preciso pensar em um padrão de comunicação que seja entendido entre ambas as partes. Neste trabalho, foi selecionado o padrão REST (*Representational State Transfer*), que é um estilo de arquitetura baseada em serviços e usa o HTTP (*HyperText Transfer Protocol*) para fazer chamadas entre máquinas, não requer muitos recursos, pois trabalha com os padrões já definidos pelo HTTP, O método de comunicação é chamado de mensagem REST e tem o JSON (*JavaScript Object Notation*) como modelo de troca de mensagem [Fielding 2000].

3. Solução Proposta

A arquitetura SOA para sistemas IoT proposta consiste em um servidor, um consumidor móvel e um consumidor embarcado. O servidor estará hospedado em uma máquina de pequeno porte, podendo ser um computador pessoal ou um Raspberry Pi. No servidor SOA serão disponibilizados serviços para serem consumidos por clientes embarcados em sistemas Android [Pereira and da Silva 2009] e em uma placa de desenvolvimento NodeMCU [1], ambos atendendo ao requisito de baixo custo. A seguir são descritos os componentes da arquitetura proposta.

[1] NodeMCU - http://nodemcu.com

3.1. Servidor

O GlassFish [2] é um servidor de aplicações baseado na Plataforma Java EE (*Enterprise Edition*) para o desenvolvimento e execução de aplicações e serviços web, oferecendo desempenho, confiabilidade, produtividade e facilidade. Atualmente ele se encontra na versão 4.1.1, dentre suas características, seus pontos fortes para montar um *web service* são alta disponibilidade, escalabilidade e tolerância a falhas, que são fundamentais para um servidor de produção que hospede aplicações de médio e grande porte [Luvizon 2012].

3.2. Cliente embarcado

O módulo NodeMCU é uma plataforma IoT de código aberto. Ela inclui o *firmware* que roda no ESP8266 (chip de baixo custo com o protocolo TCP/IP (*Transmission Control Protocol/Internet Protocol*) integrado) e um *hardware* baseado no módulo ESP-12 (a mais recente versão do ESP8266). O *firmware* usa a linguagem de programação Lua.

A plataforma NodeMCU foi escolhida pelo seu baixo custo e por possuir os recursos suficientes para diversas aplicações que envolvam a Internet das Coisas, como realizar uma comunicação entre sensores e um servidor. Uma outra característica importante para a escolha do NodeMCU foi a capacidade de programação utilizando o IDE (*Integrated Development Environment*) do Arduino, que permite usar uma variação da linguagem C (linguagem Wiring) [Silveira and de Lima Leite 2016].

3.3. Cliente móvel

O software cliente para a prova de conceito é uma aplicação móvel desenvolvida para a plataforma Android, o ponto forte do Android está na capacidade de fazer com que todos os dispositivos que o utilizem sigam padrões de conexões, processamento e programação, facilitando o desenvolvimento de novos aplicativos que aprimoram o seu uso [Pereira and da Silva 2009].

4. Prova de Conceito

O sistema no qual foi aplicada a proposta de IoT utilizando serviços é uma aplicação de monitoramento de um ambiente por sensores de presença e temperatura. Em cada vez que o sensor de temperatura é ativado, no projeto foi usado um intervalo definido de cinco minutos, ou em cada vez que o sensor de presença é ativado, ou seja, cada vez que ele detecta um movimento, são enviadas as informações para o servidor através de um método HTTP POST. Essas informações contêm o número do sensor que está sendo ativado e, no caso do sensor de temperatura, a temperatura em graus Celsius. A mensagem é passada no formato JSON e deve seguir os padrões determinados pelo servidor.

Ao receber a mensagem pelo POST, o servidor se encarrega de verificar se é um JSON válido, caso não seja, ele retorna uma mensagem de erro, e caso for válido, ele faz o tratamento do JSON e armazena seus dados em um banco de dados. Os dados armazenados são a data e a hora do recebimento, o número do sensor e, se for o caso, a temperatura. Cada operação é feita com um serviço diferente, há um serviço POST para o sensor de temperatura e outro para o de presença.

[2] Glassfish - Pesquisas de compiladores. http://pesquompile.wikidot.com/glassfish

O aplicativo para Android faz uma solicitação HTTP GET no servidor, que responde de acordo com a informação solicitada, assim como para o POST, o servidor também possui dois serviços GET, um para temperatura e outro para a presença. Ao fazer essa solicitação, é retornado um JSON com todo o histórico de eventos armazenados do sensor ou temperatura, então o Android faz o tratamento desse dados e mostra na tela o último evento armazenado para temperatura ou presença.

4.1. Servidor de Serviços

Para o gerenciamento da infraestrutura de *software*, o servidor REST foi implementado utilizando a aplicação GlassFish associado à *IDE NetBeans*.

Os serviços que foram desenvolvidos e disponibilizados no servidor são:

- Armazenar dados do sensor de presença: um método POST que recebe um JSON com o número do sensor e armazena no banco de dados esse número e a data e hora do recebimento;
- Armazenar dados do sensor de temperatura: um método POST que recebe um JSON com o número do sensor e a temperatura, então armazena no banco de dados com a data e hora;
- Mostrar histórico de dados do sensor de presença: um método GET que retorna todos os eventos em que os sensores de presença foram ativados;
- Mostrar histórico de dados do sensor de temperatura: um método GET que retorna todos os eventos em que os sensores de temperatura foram ativados.

O acesso aos serviçõe são feitos através de uma URL, cada serviço tem sua própria, a estrutura da URL é definida da seguinte forma:

- *localhost* ou IP (*Internet Protocol*) da máquina local: é o endereço de IP local da rede que a máquina executando o *server* obtém;
- 8080: porta que é feita a troca de informação;
- Nome do servidor: no caso, tem o nome de ”ServerREST”;
- Application Path: caminho da aplicação que está sendo executado no servidor;
- Path: é o nome do serviço solicitado.

Na Figura 2 podem ser vistas as entradas de dados no servidor.

4.2. Cliente embarcado

Na aplicação embarcada foram utilizados uma placa NodeMCU, um sensor de temperatura e um sensor de movimento. A conexão dos sensores na placa foi feita por meio de uma *Protoboard*. Para o código que é executado pelo NodeMCU, foram utilizados alguns comandos presentes do exemplo de código ”ESP8266HTTPClient: BasicHttpClient”, disponível na IDE do Arduino.

Na função ”*loop*” é colocado o algoritmo que será executado. No presente experimento, a seguinte lógica foi empregada: A medição da temperatura é feita a cada intervalo de tempo predefinido, e a cada medição, a mensagem é enviada para o servidor pelo método POST. O sensor de movimento, ao ser ativado, envia instantaneamente uma mensagem para o servidor pelo mesmo método. Observa-se que os serviços responsáveis por cada medição (do sensor de temperatura e do de movimento) são independentes.

Feitas todas estas modificações, o programa está pronto. Ao ser enviado para a placa, o processo de conexão do módulo é iniciado, os dados dos sensores são lidos e enviados para o servidor indicado por meio do método POST. A placa e os sensores são apresentados na Figura 1.

4.3. Cliente móvel

Visando a parte que utiliza os dados disponíveis no servidor, um breve exemplo foi desenvolvido. Os serviços, para serem consumidos, necessitam de uma plataforma, no caso, é utilizada a plataforma Android, a proposta central é o desenvolvimento de um aplicativo consumidor desses serviços, que trabalhe na mesma arquitetura que o servidor, portanto, um aplicativo que faça uma requisição GET no servidor, pegue a resposta em JSON, manipule os dados e exiba a informação em forma de consulta no dispositivo. O aplicativo pode ser visto em funcionamento na Figura 3

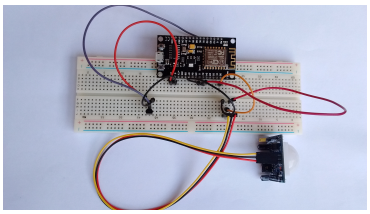

Figure 1. Prova de Conceito

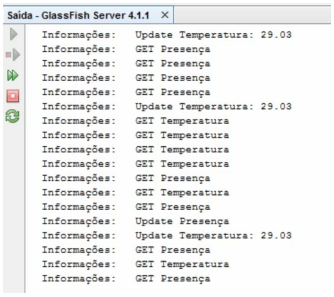

Figure 2. Banco de Dados

Figure 3. Aplicativo Android

5. Análise dos Resultados

O teste do sistema foi realizado em uma sala climatizada, portanto, era possível alterar a temperatura do ambiente para verificar a eficiência da medição realizada pelo sensor de temperatura, e se os dados provenientes dos sensores eram enviados corretamente. O sensor de presença tem sua ativação de forma binária, ou seja, está ativado ou desativado, ele contém dois potenciômetros para ajustar a distância de medição e o tempo de aviso. Os ajustes foram feitos de forma satisfatória para detectar os movimentos de uma pessoa em uma sala. O conjunto de sensores ficaram em uma posição superior ao nível médio da altura da sala, para evitar a obstrução de detecção por objetos e o sensor de temperatura foi ajustado para mandar uma medição de temperatura a cada cinco minutos.

O NodeMCU foi configurado para se conectar a uma rede *WiFi* na qual o servidor também está conectado. Então, foi ajustado o envio das informações dos sensores para o IP local da máquina do servidor, de acordo com a URL de cada serviço. No servidor, foi verificado o recebimento e registro dos dados pela saída do programa, onde foi configurado para mostrar cada vez que um serviço fosse executado. A aplicação móvel foi programada para, quando o usuário solicitar, ele faz uma GET e exibe o último evento armazenado no servidor de cada sensor. São mostrados na tela o número do sensor que foi feito o registro, a data e hora do evento e, no caso do sensor de temperatura, a temperatura.

6. Conclusão

A prova de conceito, como um todo, funcionou de forma coerente com o que foi proposto, conseguiu-se aplicar uma arquitetura orientada a serviços em dispositivos embarcados, fazendo a integração deles. Os dados do monitoramento são disponibilizados através de serviços HTTP, com mensagem em JSON, seguindo os padrões de um servidor REST, podendo ser acessado por qualquer dispositivo que tenha conhecimento da URL e trabalhe com os padrões de comunicação, sendo possível tal dispositivo fazer a manipulação dos dados da forma que ele achar necessário.

Há uma série de vantagens visíveis ao se aplicar serviços na troca de dados entre sistemas embarcados. Em comparação com o exemplo feito no projeto, sem serviços, a disponibilização dos dados da placa NodeMCU teria que ser pensada de forma independente para cada plataforma. Para o Android, por exemplo, o tipo de comunicação teria que ser feita via Bluetooth [3], e tais dados seriam visíveis apenas por um dispositivo conectado por vez, não sendo possível a criação do banco de dados em outras plataformas. Portanto, percebe-se a universalidade dos dados independentemente das plataformas, como pretendido.

A aplicação móvel pode ser aprimorada com funcionalidades que demonstrem a grande variedade de possibilidades de utilização dos dados recebidos. Por exemplo, pode ser criado um algoritmo que calcule estatísticas relacionadas à frequência de ativação do sensor de presença, assim como consultas por períodos de tempo, exibindo uma média dos dados recebidos em um determinado período de tempo. Tais estatísticas podem ser utilizadas para traçar um perfil do usuário, caracterizando-se como processamento de dados, princípios das propostas IoT e *Big Data*.

References

Ashton, K. (2009). That 'internet of things' thing. *RFiD Journal*, 22(7):97–114.

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

Ferreira, P. B. V. (2015). Arquitetura rest em smartphones android. Master's thesis.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine.

Friess, P. (2013). *Internet of things: converging technologies for smart environments and integrated ecosystems*. River Publishers.

Luvizon, J. G. (2012). Segurança e desempenho em aplicações web utilizando jaas, glassfish e postgresql.

Pereira, L. C. O. and da Silva, M. L. (2009). *Android para desenvolvedores*. Brasport.

Silveira, R. M. and de Lima Leite, S. (2016). Sistema de controle de acesso baseado na plataforma nodemcu.

[3] Arduino e Cia. http://www.arduinoecia.com.br/2014/01/comunicacao-arduino-android-com-bluetooth.html