Microcontroladores
Wireless Monitor - Aplicativo livre web para receber e mostrar dados vindos de equipamentos IOT

Átila Camurça Alves

[1]Instituto Federal do Ceará (IFCE)

Abstract. *The Wireless Monitor app has the goal to allow embed systems developer to send data to the cloud collected by his IOT device and preview in a browser.*

Resumo. *O aplicativo Wireless Monitor tem o objetivo de permitir que desenvolvedores de sistemas embarcados possam enviar para a nuvem os dados obtidos por seu equipamento IOT e visualizá-los no navegador.*

1. Introdução

O aplicativo Wireless Monitor tem o objetivo de permitir que desenvolvedores de sistemas embarcados possam enviar para a nuvem os dados obtidos por seu equipamento IOT e visualizá-los no navegador.

Em equipamentos que usam microcontroladores não existe a figura de um monitor em que se possa verificar a saída dos comandos e acompanhar sua execução, existem apenas saídas seriais ou placas wifi embutidas. Daí surge a necessidade de criar sistemas que possam recolher os dados enviados por esses equipamentos e mostrá-los de forma apropriada.

2. Objetivos

O objetivo principal é fornecer uma *api* leve e simples, visto que equipamentos IOT são limitados, para enviar e receber informações da nuvem.

Para que haja melhor intercâmbio das informações tanto partindo do equipamento IOT quanto chegando o protocolo de comunicação escolhido foi o JSON, que segundo Douglas Crockford é um formato leve e de linguagem independente para troca de informações [Crockford 2015].

Tendo isso em vista, vejamos os passos para criar um projeto.

2.1. Cadastro do desenvolvedor

O desenvolvedor inicialmente deve fazer um cadastro simples na ferramenta. Esse cadastro irá criar para ele uma `api_key`, ou seja, uma chave única no formato UUIDv4.

2.2. Criar um *Monitor*

Um *Monitor* é um componente interno do sistema criado pelo desenvolvedor de acordo com sua necessidade, é o instrumento que caracteriza os dados coletados e os apresenta na interface web.

Imagine que o desenvolvedor queira medir a temperatura de um ambiente e acompanhar suas variações. Para isso ele deve criar um *Monitor* de Temperatura, que apenas recebe um valor a um certo intervalo de tempo. Dessa forma o desenvolvedor pode acompanhar as variações ou ainda ver em forma de gráfico um conjunto de variações de um período de tempo anterior.

Da mesma forma que uma chave UUID é criada para o desenvolvedor, uma chave é criada para o Monitor - `monitor_key`.

2.3. Autenticação do equipamento

Para autenticar e identificar o desenvolvedor e seu *monitor* é preciso enviar a `api_key` e a `monitor_key` via método *POST* para o *endpoint* `/api/authenticate`. Em caso positivo o sistema irá retornar um *token*. Esse *token* servirá para qualquer troca de informações futuras entre o equipamento IOT e o sistema. Esse método de autenticação é chamado de JWT ou *JSON Web Token*, um padrão internacional *RFC 7519* para intercâmbio de dados entre entidades.

Após ter o *token* o desenvolvedor deve passá-lo através da *Header HTTP* denominada *Authorization* usando *schema Bearer*. Algo do tipo:

```
Authorization: Bearer <token>
```

Um *token* é formado pelas seguintes informações:

- *Header*
- *Payload*
- *Signature*

Essa é uma forma segura e com pouco custo de memória. Além de ser uma forma de autenticação *stateless*, em que não são usadas sessões e nem mesmo *cookies*.

2.4. Envio dos dados

Além do cabeçalho contendo o *token* o usuário deve passar os valores coletados pelo equipamento e enviar para o sistema. Para isso ele deve enviar uma requisição *POST* para o *endpoint* `/api/send`, com o atributo `data` contendo um JSON com os dados.

No exemplo do *Monitor* de temperatura é necessário enviar apenas o valor, algo do tipo:

```
{
    "value": 23.89
}
```

Sumarizando o código final seria algo como:

```
HTTPClient http;
http.begin("https://wireless-monitor.site/api/send");
http.addHeader("Content-Type", "application/json");
http.addHeader("Authorization", "Bearer <token>");
http.POST("data={value:23.89}");
http.writeToStream(&Serial);
http.end();
```

2.5. Visualização dos dados

Após captar e enviar dados do IOT para a nuvem é possível acompanhar os resultados pelo sistema. A forma de visualização será como mostra a Figura 1

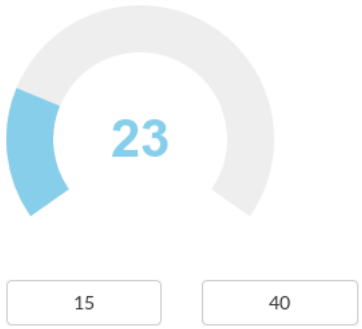

Figura 1. Visualização dos dados na web

3. Justificativa

Sendo um aplicativo de código-fonte licenciado pela GPLv3 poderá ser usado tanto para professores e alunos de cursos superiores e técnicos para estudo de microcontroladores, sistemas embarcados e afins, como para empresas ou pessoas que queiram interagir com seus equipamentos pessoais.

A linguagem de programação escolhida foi o PHP, a qual é fácil de aprender, normalmente lecionada em cursos superiores e técnicos e de hospedagem barata.

Outra característica a ser levada em conta é a forma de autenticação. Uma autenticação convencional envolve a troca de *cookies* entre servidor e cliente, além de espaço em disco para guardar tais informações. Em sistemas IOT que se supoem que possam crescer de forma rápida, ou seja, o número de equipamentos pode aumentar, é necessário um sistema de autenticação capaz de ser escalável mesmo em condições limitadas. Para isso foi utilizado o padrão JWT (ou *JSON Web Tokens*), que é um padrão aberto (RFC 7519 [Michael B. Jones and Sakimura 2015]) que define uma maneira compacta e auto-contida de transmitir de forma segura informações entre pares através de um objeto JSON [JWT 2016]. Esta informação pode ser verificada e confirmada pois é assinada digitalmente. Informações JWT podem ser assinadas usando um segredo (com o algoritmo HMAC [Hugo Krawczyk and Canetti 1997]) ou um par de chave pública e privada usando RSA [Jonsson and Kaliski 2003].

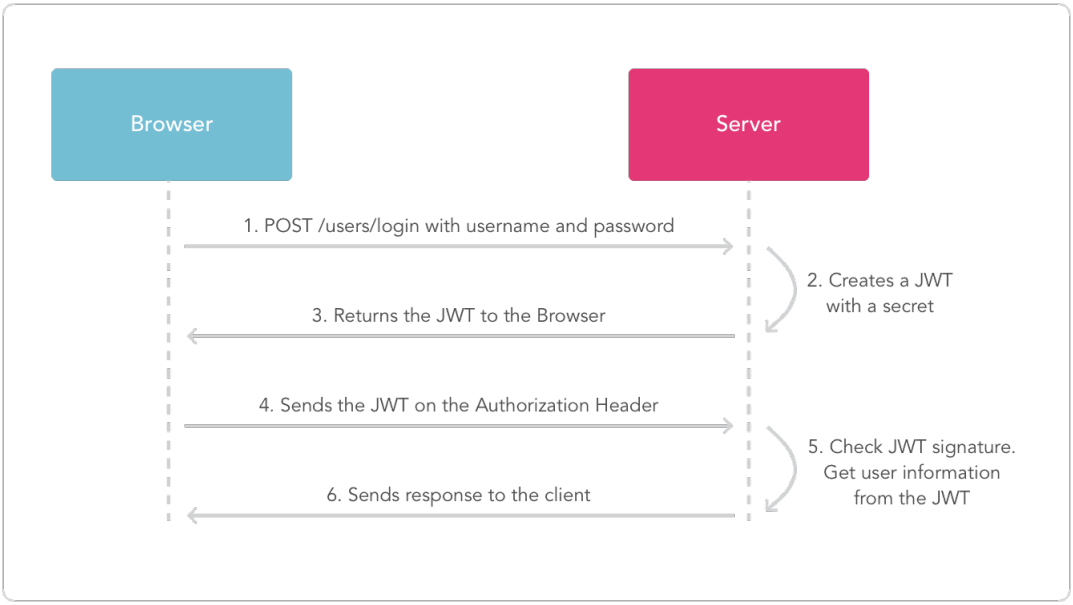

Figura 2. Diagrama do processo de autenticação - Fonte: https://cdn.auth0.com/content/jwt/jwt-diagram.png

4. Revisão Teórica

Muitas são as soluções de monitoramento de equipamentos IOT, grandes empresas com Oracle, Amazon, Google, Microsoft; além de outras soluções livres como Kaa, ThingSpeak, macchina.io, SiteWhere [Postscapes 2016].

O grande desafio é permitir a extensão da ferramenta para necessidades específicas. Ferramentas com o Kaa permitem criar módulos próprios, sistemas de análises

e modelo de dados, fazendo com que a ferramenta se adapte ao que você precisa [Kaa 2014].

De forma semelhante outras ferramentas como macchina.io oferecem opções de criar *bundles* [Macchina.io 2016], o ThingSpeak oferece opção de criar *apps*, que podem envolver visualização em gráficos e tomada de decisões [ThingSpeak 2016].

Nesse sentido a ferramenta proposta possui um sistema de plugins, que são desenvolvidos como *Laravel Packages* [Laravel 2016]. Cada nova funcionalidade é criada através da ferramenta *Laravel* e pode ser desenvolvida e habilitada localmente.

A proposta é ter uma tela de acompanhamento dos dados captados do equipamento e a visualizaçãos ser específica. A documentação em português do brasil para criar um novo plugin pode ser encontrada em `https://sanusb-grupo.github.io/wireless-monitor/pt-br/plugin-development.html`.

5. Cronograma

Tarefas	Semana 1	Semana 2	Semana 3	Semana 4
Protótipo Inicial	X			
Programação	X	X	X	
Testes			X	X
Relatório	X			X

Tabela 1: Cronograma

Referências

[Crockford 2015] Crockford, D. (2015). JSON. https://github.com/douglascrockford/JSON-js/blob/master/README. [Online; accessed 13-September-2016].

[Hugo Krawczyk and Canetti 1997] Hugo Krawczyk, M. B. and Canetti, R. (1997). HMAC: Keyed-Hashing for Message Authentication. https://tools.ietf.org/html/rfc2104. [Online; accessed 13-September-2016].

[Jonsson and Kaliski 2003] Jonsson, J. and Kaliski, B. (2003). Public-Key Cryptography Standards (PKCS) 1: RSA Cryptography Specifications Version 2.1. https://tools.ietf.org/html/rfc3447. [Online; accessed 13-September-2016].

[JWT 2016] JWT (2016). Introduction to JSON Web Tokens. https://jwt.io/introduction/. [Online; accessed 13-September-2016].

[Kaa 2014] Kaa (2014). Dev center - Complete application. http://www.kaaproject.org/platform/#complete-application. [Online; accessed 13-September-2016].

[Laravel 2016] Laravel (2016). Package Development. https://laravel.com/docs/5.2/packages. [Online; accessed 13-September-2016].

[Macchina.io 2016] Macchina.io (2016). Bundles Overview. http://macchina.io/docs/00200-OSPBundles.html. [Online; accessed 13-September-2016].

[Michael B. Jones and Sakimura 2015] Michael B. Jones, J. B. and Sakimura, N. (2015). JSON Web Token (JWT). https://tools.ietf.org/html/rfc7519. [Online; accessed 13-September-2016].

[Postscapes 2016] Postscapes (2016). IoT Cloud Platform Landscape. http://www.postscapes.com/internet-of-things-platforms/. [Online; accessed 13-September-2016].

[ThingSpeak 2016] ThingSpeak (2016). Apps. https://thingspeak.com/apps. [Online; accessed 13-September-2016].