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

Átila Camurça Alves[1]

[1]Instituto Federal do Ceará (IFCE)

Abstract. *This article presents the Wireless Monitor app that has the goal to allow embed systems developers to send data to the cloud collected by an IOT device and preview in a browser.*
In devices that use microcontrollers the figure of a monitor (screen) doesn't exist, which one can check the commands output e follow its execution, there's just serial outputs and embed wi-fi boards. From there rises the need to create systems that could gather data sended by this devices e show them in a appropriate way, there is, a web browser.

Resumo. *Este artigo apresenta o aplicativo Wireless Monitor que tem o objetivo de permitir que desenvolvedores de sistemas embarcados possam enviar para a nuvem os dados obtidos por equipamento IOT e visualizá-los no navegador.*
Em equipamentos que usam microcontroladores não existe a figura de um monitor (tela), em que se possa verificar a saída dos comandos e acompanhar sua execução, existem apenas saídas seriais ou placas wi-fi embutidas. Daí surge a necessidade de criar sistemas que possam recolher os dados enviados por esses equipamentos e mostrá-los de forma apropriada, isto é, um navegador de internet.

1. Introdução

Com a crescente adoção de equipamentos IOT, como por exemplo o Raspberry Pi que em quase 5 anos vendeu 10 milhões de unidades pelo mundo [Upton 2016], para monitoramento de sensores e acionamento de cargas, cresce também a necessidade de ambientes de acompanhamentos de tais medições. Para isso uma das melhores formas é usar a nuvem para fazer o armazenamento, já que uma das caracteristicas dos equipamentos IOT é o acesso a internet. Para atender essa necessidade surge a ideia de criar um aplicativo web e livre que possa captar informações destes dispositivos e que o acesso possa acontecer em qualquer lugar.

2. Objetivos

O objetivo principal do Wireless Monitor é fornecer uma *api* leve, simples e segura, 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].

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].

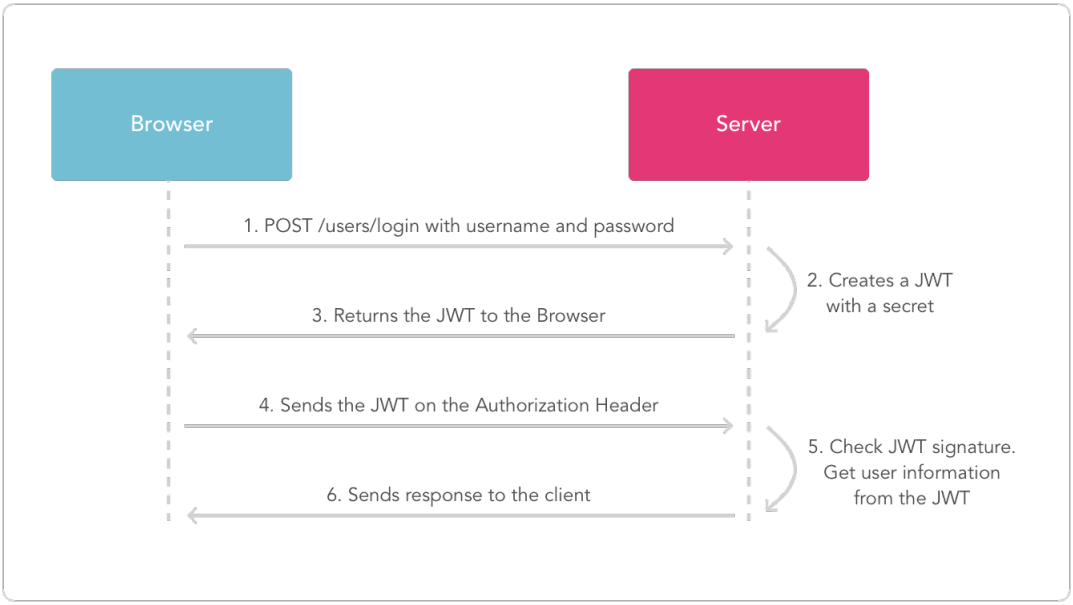

Figura 1. Diagrama do processo de autenticação - Fonte: https://cdn.auth0.com/content/jwt/jwt-diagram.png

4. Revisão Teórica

Muitas são as soluções de monitoramento de equipamentos IOT, desde grandes empresas como Oracle, Amazon, Google, Microsoft; até 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`.

4.1. Comparativo com outras ferramentas livres

Aplicativo	Ambiente do Servidor	Suporte a plugins	SDK
Kaa	Java	Sim	Sim
macchina.io	C++/NodeJS	Sim	Sim
SiteWhere	Java	Sim	Sim
ThingSpeak	Ruby	Sim	Sim
Wireless Monitor	PHP	Sim	Não

5. Arquitetura

É necessário um Servidor, um equipamento IOT, seja ESP8266 ou Raspberry Pi com Arduino; e um navegador de internet no cliente.

A arquitetura segue o modelo da Figura 2.

6. Aplicação

Vejamos um passo a passo de como o aplicativo funciona.

6.1. 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 UUID 4 [Paul J. Leach and Salz 2005].

6.2. 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.

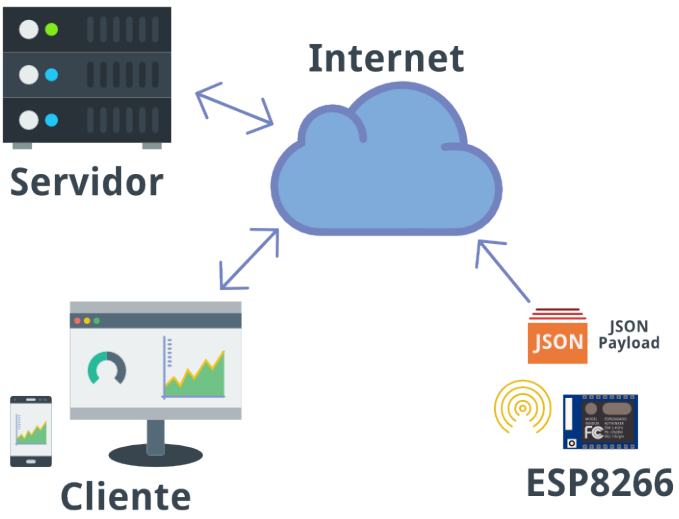

Figura 2. Arquitetura

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`.

6.3. 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.

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*.

6.4. 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
}
```

6.5. Descrição dos componentes

Neste tópico são descritos os componentes e as ferramentas utilizadas para o desenvolvimento e uso do plugin de Temperatura.

6.5.1. Sistema embarcado linux

Foi utilizado a plataforma Raspberry Pi como sistema embarcado, que irá servir para cominucação com o Servidor e o dispositivo de captura de temperatura.

6.5.2. Microcontrolador

A plataforma Arduino foi escolhida para servir de ponte entre o componente de medição de temperatura e o sistema embarcado.

6.5.3. Sensor de temperatura

Como sendor de temperatura foi usado o LM35 da Texas Instruments. A série LM35 é composta de dispositivos de circuito integrado para medição de temperatura com a tensão de saída linearmente proporcional a temperatura em graus Celcius. O sensor LM35 tem a vantagem sobre sensores de temperatura linear calibrados em Kelvin, devido a não ser necessário subtrair uma alta tensão constante da saída para obter uma escala conveniente [Instruments 2016].

6.6. Ambiente de execução

Para esse exemplo o ambiente de execução escolhido foi o NodeJS, que é um envólucro (*wrapper*) do ambiente de execução JavaScript de alta performance chamado V8 usado no navegador Google Chrome. O NodeJS permite que o V8 funcione em contextos diferentes do browser, principalmente fornecendo APIs adicionais que são otimizadas para casos específicos [Hughes-Croucher and Wilson 2012]. Por exemplo no caso de equipamentos IOT é perfeito, pois se trata de um dispositivo orientado a eventos, assim como o NodeJS.

Para auxiliar na conversação entre o NodeJS e o Arduino foi usado a ferramenta Johnny-Five, uma plataforma livre Javascript para Robôs e IOT [Waldron 2012].

6.7. Princípios de execução

O NodeJS deve ser instalado no Raspberry Pi já que possui suporte a arquitetura ARM. Um projeto NodeJS deve ser criado tendo como dependências o Johnny-Five e uma biblioteca de resquisições HTTP, como por exemplo `request` [Request 2016]. Dessa forma o Johnny-Five se encarregará de se comunicar com o Arduino requisitando a temperatura

do componente LM35. Com a resposta em mãos o NodeJS irá enviar as medições ao Servidor através da biblioteca `request`.

O código fonte deste exemplo pode ser encontrado num repositório do GitHub [Alves 2016].

6.8. 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 3.

Figura 3. Visualização dos dados na web

7. Conclusão

A partir de ferramentas livres é possível sim criar ambientes de alta qualidade para monitoramento de dispositivos IOT. Tanto porque grande parte das ferramentas livres são estáveis e bem testadas, quanto a liberdade de poder customizar para que a ferramenta atenda sua necessidade, não o contrário.

Um passo importante e que intimida um pouco é a forma de autenticação. JWT é uma tecnologia muito recente e utiliza técnicas pouco convencionais para o público iniciante, mas temos que levar em conta que junto com o aumento do uso de equipamentos ligados a internet vem a necessidade de segurança na comunicação. Uma falha de segurança que se tornou comum nessas situações é chamada de *man-in-the-middle*, que pode ser definida como "Uma falha de segurança em um computador em que um usuário malicioso intercepta - e possivelmente altera - dados trafegando em uma rede" [Wordspy 2002]. Esse erro pode ocorrer simplesmente porque os vínculos entre aparelhos e criptografias de proteção cedidas por um padrão normalmente não foram implementadas corretamente, como por exemplo em travas eletrônicas que usam Bluetooth Low Energy (BLE) [Spring 2016].

Uma prática comum para autenticação de IOTs é a criação de tokens randômicos para identificar o usuário e o dispositivo, entretanto essa técnica facilita o ataque *man-*

in-the-middle. Nessa linha o uso do JWT possui vantagens quando comparado com um token randômico:

- Chaves API randômicas não dizem nada a respeito do usuário, enquanto JWTs contém informações e metadados que descrevem a identidade do usuário; contém também uma validade por um período de tempo ou domínio.
- JWT não obriga a necessidade um emissor de token centralizado ou autoridade de revogação de token.
- É compatível com Oauth2 [Atwood et al. 2012].
- Dados do JWT podem ser inspecionados.
- JWTs possuem controles de expiração [Romero 2015].

Por fim o passo seguinte seria permitir o envio de comandos do navegador para o dispositivo, podendo assim controlar algumas funcionalidades remotamente como o acionamento de cargas, disparo de relés, entre outras funções.

Referências

[Alves 2016] Alves, Á. C. (2016). Sensor de temperatura usando plataforma IOT Wireless Monitor. https://github.com/atilacamurca/wm-sensor-temperature. [Online; accessed 23-September-2016].

[Atwood et al. 2012] Atwood, M., Balfanz, D., Bounds, D., Conlan, R. M., Cook, B., Culver, L., de Medeiros, B., Eaton, B., Elliott-McCrea, K., Halff, L., Hammer, E., Laurie, B., Messina, C., Panzer, J., Quigley, S., Recordon, D., Sandler, E., Sergent, J., Sieling, T., Slesinsky, B., and Smith, A. (2012). The OAuth 2.0 Authorization Framework. https://tools.ietf.org/html/rfc6749. [Online; accessed 23-September-2016].

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

[Hughes-Croucher and Wilson 2012] Hughes-Croucher, T. and Wilson, M. (2012). Node: Up and Running. http://chimera.labs.oreilly.com/books/1234000001808/index.html. [Online; accessed 21-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].

[Instruments 2016] Instruments, T. (2016). LM35 Precision Centigrade Temperature Sensors. http://www.ti.com/lit/ds/symlink/lm35.pdf. [Online; accessed 20-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].

[Paul J. Leach and Salz 2005] Paul J. Leach, M. M. and Salz, R. (2005). A Universally Unique IDentifier (UUID) URN Namespace. https://tools.ietf.org/html/rfc4122. [Online; accessed 17-September-2016].

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

[Request 2016] Request (2016). Simplified HTTP client. https://github.com/request/request. [Online; accessed 21-September-2016].

[Romero 2015] Romero, M. I. (2015). PHP Authorization with JWT (JSON Web Tokens). https://www.sitepoint.com/php-authorization-jwt-json-web-tokens/. [Online; accessed 23-September-2016].

[Spring 2016] Spring, T. (2016). Bluetooth Hack Leaves Many Smart Locks, IoT Devices Vulnerable. https://threatpost.com/bluetooth-hack-leaves-many-smart-locks-iot-devices-vulnerable/119825/. [Online; accessed 23-September-2016].

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

[Upton 2016] Upton, E. (2016). Ten millionth Raspberry Pi, and a new kit. https://www.raspberrypi.org/blog/ten-millionth-raspberry-pi-new-kit/. [Online; accessed 17-September-2016].

[Waldron 2012] Waldron, R. (2012). Johnny-Five: the JavaScript Robotics IoT Platform. http://johnny-five.io/. [Online; accessed 21-September-2016].

[Wordspy 2002] Wordspy (2002). man in the middle attack. http://wspy.ws/874. [Online; accessed 23-September-2016].