# Free and customizable web application for Internet of Things devices monitoring

Átila Camurça Alves [1], Jessé Perbelini Coutinho [2], Vinícius Ferreira da Silva [3], Sandro César Silveira Jucá [4] e Renata Imaculada Soares Pereira [5]

*[1] camurca.home@gmail.com. [2] jessecdm@gmail.com. [3] viniciusdasilva.ti@gmail.com. [4] sandrojuca@ifce.edu.br. Instituto Federal do Ceará – Campus Maracanaú, Eixo de Computação. [5] renata@dee.ufc.br. Instituto Federal de Alagoas – Campus Arapiraca, Eixo de Controle e Processos Industriais.*

## ABSTRACT

This paper presents the design and implementation of the free and customizable web application for Internet of Things (IoT) named Wireless Monitor. Wireless Monitor main objective is to provide a solution for online and real-time secure data visualization. The user can create a personalized web ambient using Wireless Monitor to monitor IoT sensor devices. The data collected is stored on a cloud server databank and can be viewed as real-time charts. The proposed Wireless Monitor has been developed to be extensible and can be adapted to different types of sensors, thanks to the system of plugins. Its minimal set of endpoints for exchanging information between the IoT device and the server makes the development simplified, without being limited, thanks to the JSON information exchange protocol. Two different practical experiments were carried out and the obtained results are shown. The first one uses Raspberry Pi, Arduino and a LM35 temperature sensor; and the second one replaces both embedded systems with one ESP 8266 development board.

Keywords: Internet of Things. Monitoring. Data acquisition. Free software.

## RESUMO

*Este artigo apresenta a criação, o design e a implementação de um aplicativo gratuito e personalizável para Internet das Coisas (IoT, do inglês Internet of Things) denominado Wireless Monitor. O objetivo principal do Wireless Monitor é fornecer uma solução para aquisição e monitoramento de forma segura de dados de sensores, de forma online e em tempo real. O usuário pode criar um ambiente web personalizado através do Wireless Monitor com o objetivo de monitorar dispositivos IoT. Os dados coletados são armazenados em um banco de dados do servidor na Nuvem e podem ser visualizados em forma de gráficos em tempo real. O Wireless Monitor proposto foi desenvolvido para ser extensível e pode ser adaptado para diferentes tipos de sensores, graças ao sistema de 'plugins'. Seu conjunto mínimo de 'endpoints' para troca de informações entre o dispositivo IoT e o servidor torna o desenvolvimento simplificado, porém sem limitações, graças ao protocolo de troca de informações JSON. Dois experimentos práticos diferentes foram realizados e os resultados obtidos são mostrados. O primeiro utiliza Raspberry Pi, Arduino e um sensor de temperatura LM35; e o segundo, substitui ambos os sistemas embarcados por uma placa de desenvolvimento ESP 8266.*

Palavras-chave: *Internet da Coisas. Monitoramento. Aquisição de Dados. Software Livre.*



SUBMETIDO 13/05/2019 > APROVADO 23/09/2019

## 1 Introduction

With the increasing adoption of the Internet of Things (IoT) in sensor monitoring systems and load driving, there is a growing need for monitoring environments for such measurements. To do this, one of the best ways is to use the Cloud — on-demand computing resource through the Internet (QIU et al, 2019) — for storage, since one of the characteristics of IoT devices is access to the Internet.

Today, the Internet of Things is a term well known in the media for presenting a real revolution in the environments in which it is found, in order to transform these environments, making them intelligent by bringing together a network of physical objects equipped with electronic components such as software and sensors, which collect and exchange data between themselves and with the user.

To meet this need, Wireless Monitor was developed, as a free and web-based application for acquisition and monitoring of sensor data in which monitoring can be done from any device through an Internet browser.

IoT devices are "things" with an Internet connection that can interconnect and communicate with each other (REVELL, 2013). As an example, Raspberry Pi is used in many ways and in different areas, such as education (KAMBOJ; KRISHNA; REDDY, 2018).

Also essential for an IoT system is the performance of objects connected to data transmission securely, processing through cloud computing (computer storage services and servers connected to the Internet) and other intelligent computing, data processing and analysis technologies (ASEMANI; ABDOLLAHEI; JABBARI, 2019).

The main goal of Wireless Monitor is to provide a simple, and secure API (Application Programming Interface), since IoT equipment are limited regarding its resources, to send and receive information from the Cloud.

In order to have a better exchange of information in receiving and sending the data, the communication protocol chosen was JSON, where this is a lightweight and independent language format for information exchange (LIU et al., 2019).

The present paper is a continuation of a work already started as can be seen in (ALVES; JUCÁ, 2017) with some adaptations and improvements that are described throughout this article.

## 2 Bibliographic Review

There are many IoT equipment monitoring solutions. We can mention the platforms: Oracle IoT, AWS IoT, Google Cloud IoT, and Microsoft Azure IoT Suite, that are developed by major companies such as Oracle, Amazon, Google, and Microsoft, respectively; and even free solutions such as Kaa and SiteWhere. Table 1 shows a comparison between the Wireless Monitor proposed and similar ones. The big challenge is to allow the tool to be extended to specific needs. Tools like Kaa allow you to create your own modules, analysis systems, and data model, making the tool fit your need (KHARBOUCH et al, 2018).

**Table 1 –** Comparison between free similar systems.

| Systems | Server environment | Plugins support | SDK | Memory | CPU | HD | Database | Hosting Price |
|---|---|---|---|---|---|---|---|---|
| Kaa | Java | Yes | Yes | 4 GB | 4 | 10 GB | - | 20 USD/Month |
| SiteWhere | Java | Yes | Yes | 16 GB | 4 | 100 GB | - | 80 USD/Month |
| Wireless Monitor | PHP | Yes | Yes | 2 GB | 1 | 5 GB | PostgreSQL | 10 USD/Month |

*Source:* Elaborated by the authors

Similarly, other tools like macchina.io offer options to create bundles, and ThingSpeak offers the option to create apps (HILL, 2019), both of which can involve visualization in graphics and decision making.

In Cantanhede and Silva (2014), a system for monitoring in hospital environments is proposed where the system aims to automate the process of collecting and processing information such as monitoring of temperature and humidity, for example where the user can program the way of using the system according to its necessity and scenario.

Shah and Mishra (2016) has developed a customized wireless monitoring platform to monitor temperature and relative humidity. In the system,

the data is sent to the receiver node and this data is monitored and recorded in excel on a desktop through a graphical interface created in LabVIEW.

Based on the concepts of Internet of Things (IoT), a residential automation system for the control and monitoring of electrical devices in Oliveira (2019) was developed. The device measures the current and alternating voltage of equipment connected to the electric grid, as well as to carry out the approximate calculation of the power consumed, in addition to making the charging of loads. The user interface is performed using ThingSpeak, where device data is monitored, stored and exported to a worksheet, and then an estimate is made for the power consumed.

In this sense, the proposed tool has a system of plugins, which are developed as Laravel Packages (BLANKENSHIP, 2019). Each new feature is created through the Laravel tool and can be developed and enabled locally.

The proposal is to have a monitoring screen of the data collected from the equipment that can be viewed in a specific way. The documentation in Brazilian Portuguese to create a new plugin can be found in Alves (2016a).

A Software Development Kit (SDK) is designed to help developers use Wireless Monitor in the JavaScript language, which should be used with NodeJS or directly in the browser. Source code and documentation can be found in a GitHub in Alves (2017a) repository.

Applications such as Kaa and SiteWhere run in Java environment, which makes deployment expensive because hosting costs more compared to hosting PHP applications, as well as processing power must be high since such an environment requires a more robust server.

## 3 Methodology

As a GNU Public License (GPLv3) licensed source code application, Wireless Monitor can be used by teachers and undergraduates, or technicians for the study of microcontrollers, embedded systems, and related fields, as well as for business and designer purposes. and enthusiasts as can be seen in Alves (2016b). The programming language used was PHP, originally derived from Personal Home Page Tools, and now means PHP: Hypertext Preprocessor, usually taught in both undergraduate and technical course levels, with cheap hosting in relation to other languages such as Java or Ruby.

Wireless Monitor is hardware-independent, as it is just an aggregator of values transmitted from any device, not necessarily IoT devices. It would be possible, for example, to create an Android application that will read its own sensors and send it to Wireless Monitor.

Another feature to consider is the form of authentication. Conventional authentication involves exchanging of cookies between the server and the client, which implies disk space to store such information. In IoT systems, which are assumed to grow rapidly, that is, the number of devices can increase, an authentication system capable of scalability is required, even under limited conditions. The JWT (JSON Web Tokens) standard, which is an open standard (RFC 7519) (JONES; BRADLEY; SAKIMURA, 2015), is used to define a compact and self-contained way of securely transmitting information between peers through a JSON object (PEYROTT, 2016). This information can be verified and confirmed because it is digitally signed. JWT information can be signed using a secret, with the HMAC (Hash-based Message Authentication Code) algorithm (KRAWCZYK; BELLARE; CANETTI, 1997) or public and private key pair using RSA (JONSSON; KALISKI, 2003). The authentication process via JWT can be seen in Figure 1 (next page).

One of the most common practices for IoT's authentication is the creation of random tokens to identify the user and the device, however, this technique facilitates the Man-In-The-Middle (MITM) attack.

- This attack can be defined as "a security flaw on a computer in which a malicious user intercepts — and possibly alters — data over a network" (YADAV; VENKATESAN; VERMA, 2018). In this sense the use of JWT has advantages when compared to a random token (ROMERO, 2015):

- Random API keys do not say anything about the user, whereas JWTs contain information and metadata that describe the user's identity;

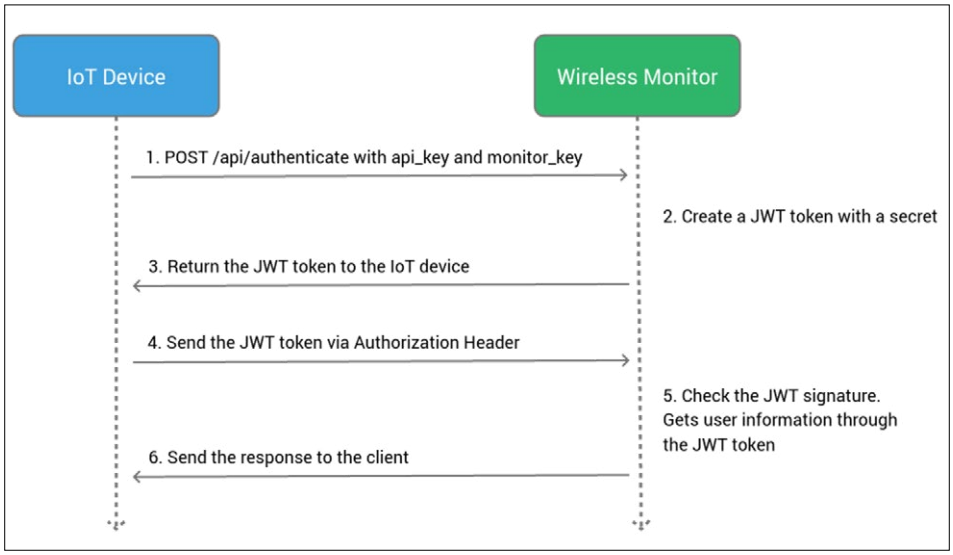
- JWT does not require the need for a centralized token issuer or token revocation authority;

- It is compatible with Oauth2 (HARDT, 2012);

- JWT data can be inspected;

- JWTs have expiration controls, contains a validity for a period or a maximum number of requests.

**Figure 1 –** Authentication process diagram via JWT.




*Source:* Elaborated by the authors.

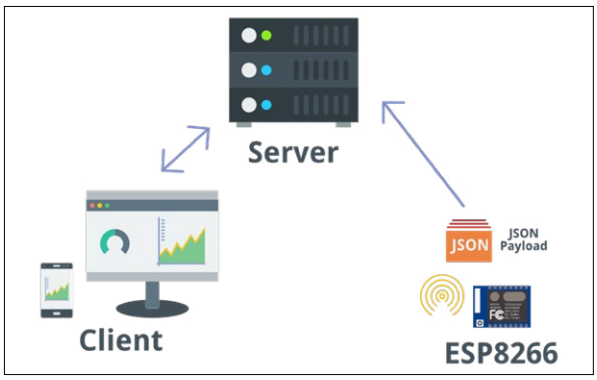
It is recommended to use the SSL protocol in the communication between the device and the server through the HTTPS protocol (RESCORLA, 2000). Because, even though JWT is a protocol that generates encrypted keys, it can also be used in MITM attacks. It is true that JWT keys have expiration dates and expire on a certain number of requests, but if the key is intercepted by a third party, it can impersonate the equipment and transmit fictitious data. However, if the information is transmitted using HTTPS, this type of attack becomes more difficult because the transmission of the information is encrypted end-to-end. In Porambage et al (2014), the authors declare that a secure communication channel establishment is crucial because IoT devices could carry private information that should not be intercepted.

To ensure that the submitted data conforms to the formatting expected, Wireless Monitor uses the JSON Schema, a vocabulary that allows annotating and validating JSON documents (JUCÁ; PEREIRA, 2018). For each plug-in, there is an associated JSON Schema, which indicates the fields the Wireless Monitor expects, as well as their types. This ensures that the data sent makes sense for the correct assembly of graphs and tables, in addition to avoiding erroneous data transmission, either due to the programmer's error or due to posting failures.

## 4 4 Architecture and Operating Principle

The architecture of Wireless Monitor follows the model of Figure 2, which consists of an IoT device that communicates with the web server through the Internet by sending a payload in the JSON format and the client request the data stored on the Cloud server displays in the browser using any computational device like desktop, notebook or mobile phone with Internet access.

**Figure 2 –** Wireless Monitor architecture.




*Source:* Elaborated by the authors.

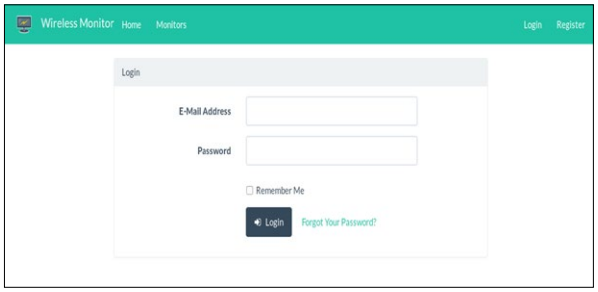
On the server there must be a Wireless Monitor instance, which works in conjunction with a PostgreSQL database, where all the data collected is stored. PostgreSQL is an ORDBMS (Object-relational Database Management System) based on POSTGRES

version 4.2, developed at the University of California, Berkeley, in the Department of Computer Science.

POSTGRES pioneered many concepts that only became available in some commercial database systems years later (MILANE, 2018).

The developer should initially make a simple registry in Wireless Monitor as can be seen in Figure 3. This registry will create for him an API key, that is, a unique key in the format UUID 4 (LEACH; MEALLING; SALZ, 2005).

**Figure 3 –** Login screen.




*Source:* Elaborated by the authors.

## 4.1 Creating a personalized Monitor

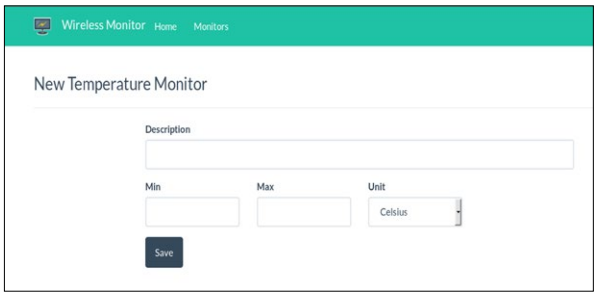
A monitor is an internal component of the system created by the developer according to his needs, it is the instrument that characterizes the data collected and presents them in the web interface.

Imagine that the developer wants to measure the temperature of an environment and monitor its variations. To do this, it must create a monitor of temperature, which only receives a value in time intervals.

In this way, the developer can monitor the sensor and its variations as well as graphically view a set of variations from an earlier period.

In the same way as creating a UUID key for the developer, a UUID key is created for monitor - monitor key. The information you need to create a temperature monitor can be seen in Figure 4.

**Figure 4 –** Creating a new temperature monitor.




*Source:* Elaborated by the authors.

## 4.2 Device Authentication

To authenticate and identify the developer and his Monitor, the user needs to send the API key and monitor key via HTTP POST method to the endpoint (or communication channel with the server) /api/authenticate. If the information is correct, the system returns a token. This token will serve for any future exchange of information between the IoT equipment and Wireless Monitor.

After getting the token, the developer should pass it through the HTTP Header named Authorization using Bearer schema (JONES; HARDT, 2012), in the format: Authorization: Bearer <token>

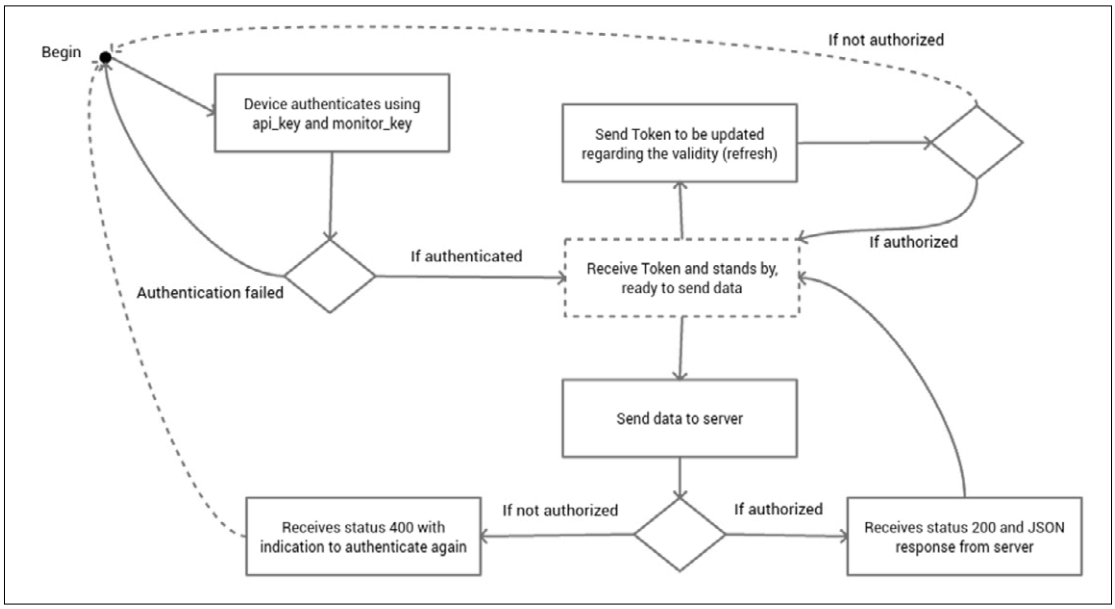
- A token is formed by the following information:
- Header;
- Payload;
- Signature.

In addition to the Header containing the token, the IoT device must pass the values collected and sent to the system. For this, it must send a POST request to endpoint /api/send, with the data attribute containing a JSON with the data collected. In the example of temperature Monitor it is necessary to send only the value, like:

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

The flowchart of the process of authentication, authorization and sending data can be seen in Figure 5.

**Figure 5 –** Flowchart: authentication, authorization, and data sending.




*Source:* Elaborated by the authors

## 5 First Experiment

In a first experiment, the Raspberry Pi platform was used as the embedded system, which communicates with the Wireless Monitor and the Arduino platform. Raspberry Pi is an IoT device capable of interacting with other devices and accessing the Internet.

The Arduino platform was chosen to serve as a bridge between the temperature sensor and the embedded system since the Raspberry Pi has only digital pins and some sensors offer only analog measurements (requiring Analog/Digital converter (ADC) pins). Arduino is a free electronic platform based on hardware and software easy to use. Arduino boards can read inputs — digital or analog sensors, buttons — and to operate outputs — driving motors, triggering LEDs (BANZI; SHILOH, 2014).

The LM35 from Texas Instruments was used as the temperature sensor. The LM35 series is composed of integrated circuit devices for measuring with the output voltage linearly proportional to the temperature in degrees Celsius. The LM35 sensor has the advantage over sensors of linear temperature, calibrated in Kelvin, because it is not necessary to subtract a constant voltage from the output to obtain a scale convenient (TEXAS INSTRUMENTS, 2017).

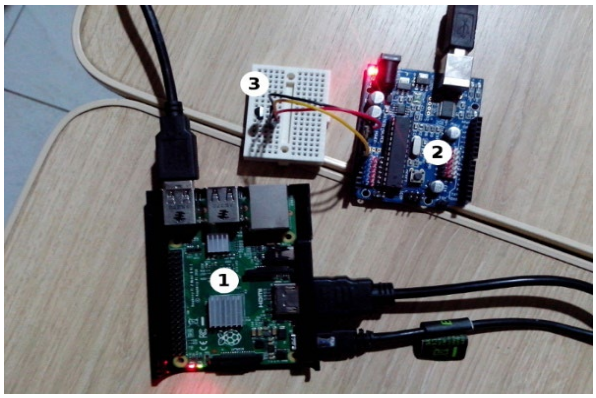
In this experiment, the execution environment chosen was the NodeJS, which is a wrapper around the high-performance JavaScript execution environment called V8, used in the Google Chrome browser. NodeJS allows V8 to work in contexts different from the browser, mainly by providing additional APIs which are optimized for specific cases (HUGHES-CROUCHER; WILSON, 2012). For example, in the case of an IoT equipment, this fits well, as it is an event-driven device, as well as the NodeJS.

To support the communication between NodeJS and Arduino, the Johnny-Five tool, a free Javascript platform for Robots and IoT was used (CRUZ; PETRUCELLI; SOTTO, 2018).

NodeJS must be installed on Raspberry Pi as it supports the ARM architecture. A NodeJS project should be created with Johnny-Five dependencies and the SDK developed to assist in integrating with the server. This way the Johnny-Five will be responsible for communicating with the Arduino requesting the temperature of the LM35 sensor. With the response received the NodeJS will send the measurements to the Server through the SDK.

The source code for this example can be found in a GitHub in repository (ALVES, 2016c) and the project assembly can be seen in Figure 6, which consists of a Raspberry Pi (1), an Arduino (2) and a LM35 temperature sensor (3).

**Figure 6 –** First experiment project assembly.



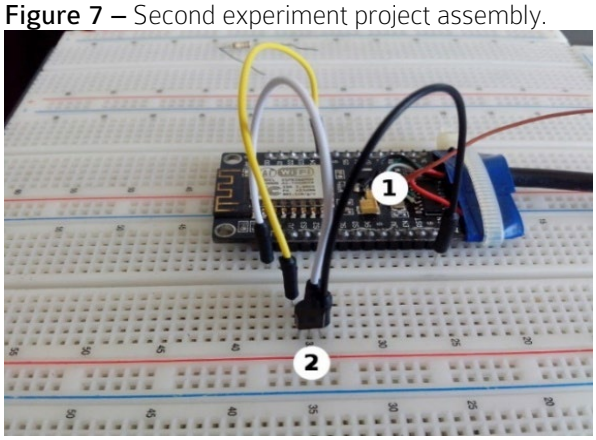

*Source:* Elaborated by the authors.

## 6 Second Experiment

In a second experiment, the device NodeMcu ESP8266 was chosen. ESP8266 is a high-performance wireless SoC (System on Chip) with integrated FTDI (Technology Integrated Chip Devices). The FTDI chip is used to power, simplify the communication process as well as the assembly process (JUCÁ; CARVALHO; PEREIRA, 2014).

The same temperature sensor LM35 from the previous experiment was used. However, the Arduino SDK with ESP8266 support was used for development, which provides a set of libraries for interaction with the network, as well as programming, compiling and assembling through the Arduino IDE (Integrated Development Environment) using the setup and loop principles, characteristic of the Arduino platform (BANZI; SHILOH, 2014). The source code of experiment 2 can be found in a GitHub repository (ALVES, 2017b) and the assembly can be seen in Figure 7, which consists of a NodeMcu with ESP8266-12F (1) and a LM35 temperature sensor (2).

**Figure 7 –** Second experiment project assembly.




*Source:* Elaborated by the authors.

## 7 Results and discussion

After capturing and sending data from an IoT device to the cloud, the user can search the results view to be displayed by the system, where the view depends on the plugin and not the data itself where the hardware is just the transmission medium.

The data visualization is shown in Figure 8 (next page). A Cloud server has been made available at "https://wm.sanusb.org" so that the tool can be used without the need for installation.

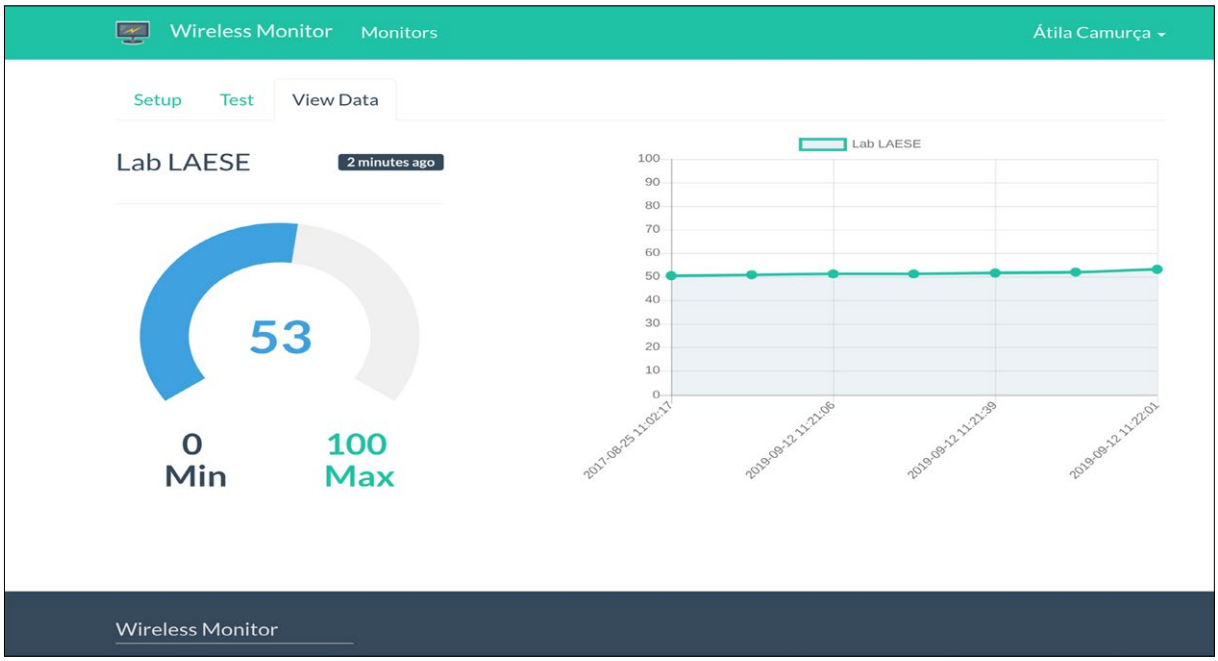
The project was developed using PHP language and Laravel framework. Both are used in the backend (server part) for the basic functionalities, that is, authentication and authorization, data visualization and management of the monitors.

For unit testing PHPUnit was used, a tool widely used in PHP projects (BERGMANN, 2001). The tests can be done locally throughout the development, however whenever a pull request (GITHUB, 2017)"title":"GitHub Help - About pull requests","type":"article"},"uris":["http://www.mendeley.com/documents/?uuid=0cf33663-a461-4ba5-9296-a668b4eb8c57"]}],"mendeley":{"formattedCitation":"(GITHUB, 2017 is done in the official project repository on GitHub, all tests are executed in a process called CI (Continuous Integration), thanks to the Travis CI tool (TRAVISCI, 2017)"title":"Travis CI - Building Pull Requests","type":"article"},"uris":["http://www.mendeley.com/documents/?uuid=bcc1acc7-a24c-4d51-b9d8-d8cedb459dd1"]}],"mendeley":{"formattedCitation":"(TRAVISCI, 2017, as can be seen in Figure 9 (next page). With this tool, it is possible to set up specific environments and test the tool in several ways, for example with several different versions of PHP. In addition to the core of the project, the plugins are also tested to ensure that the tool works completely.

Database management is done through Eloquent ORM (BEAN, 2015), which provides a single interface in which it is possible to work with several different databases by changing just the driver connection.

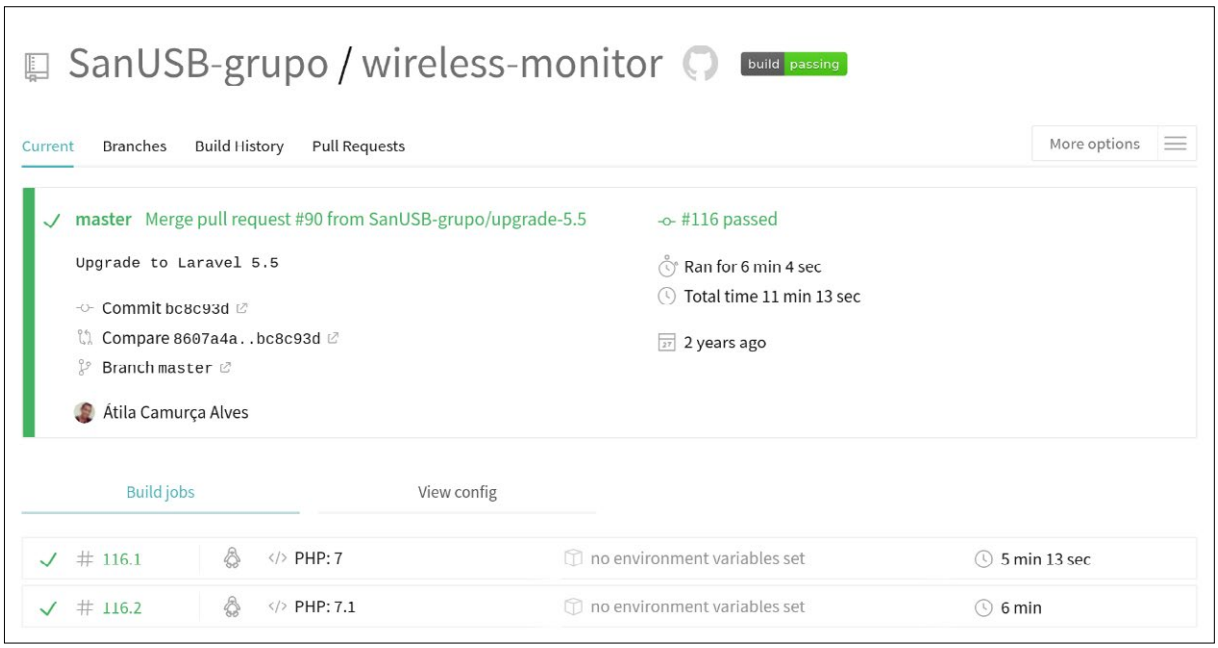
With this tool, the developer creates models that define the data structure and can interact with database tables. One of the possibilities was to allow the use of the PostgreSQL database for production environments — where robustness, stability and concurrent access to data is required, and to use the SQLite database for the test environment — where it is necessary to have a fast environment that can be created and destroyed locally, directly in RAM (ALLEN; OWENS, 2010).

**Figure 8 –** View data in Wireless Monitor.




*Source:* Elaborated by the authors.

**Figure 9 –** Unit testing and integration into the TravisCI platform.




*Source:* Elaborated by the authors.

For the front-end (part referring to the client) the tools Gulp (GULP, 2017) and Laravel Elixir (BLANKENSHIP, 2019) were used for automation of tasks involving:

- compression, add hash to the filename, avoiding unwanted cache and concatenate JavaSript and CSS files for production environments;
- creation of new plugins in Laravel Packages format;
- live reload for the development environment, thanks to Browser-Sync (BROWSERSYNC, 2017).

The documentation was created using the Gitbook (GITBOOK, 2017), that transforms files in the Markdown or AsciiDoc format into a Website in either HTML or eBook format as PDF, ePUB or Mobi. The same technology was used in the SDK documentation.

The proposed projectWireless Monitor was developed 100% based on free projects, however, a tool called demokit — application for building product demos and tutorials using web technologies — has only support for MacOS (ALVES, 2017c).

To allow the tool to work on GNU / Linux as well, a fork was created so that the screen recording tool responsible for screen recording (POUWERKERK, 2017) was replaced by the fluent-FFmpeg tool. It has the same purpose but works multi-platform, just needing to install FFmpeg which is a multiplatform multimedia framework capable of decoding, encoding and transmitting almost any kind of video and audio source (JOYARD, 2017).

Another feature was the library used for the mouse movement and control, originally written in Objective-C. It has been replaced by the RobotJS tool, that also works on all major Operating System platforms (ROBOTJS, 2017).

The modified demokit tool was responsible for creating the tutorial videos, in order to assist new users of Wireless Monitor (ALVES, 2017c). In addition to be a programmatic application, it can also be used to create videos in different languages using localization tools such as messageformat, that can switch text based on a locale context (SEXTON; ARO, 2017).

## 8 Conclusions

From free tools, it is possible to create high-quality environments for IoT device monitoring. Both because most of the free tools are stable and well tested, as well as the freedom to be able to customize the tool to meet the needs of users and developers, unlike proprietary tools.

The proposed project Wireless Monitor has been developed to be extensible and can be adapted to different types of sensors, thanks to the system of plugins. Its minimal set of endpoints for exchanging information between the IoT device and the server makes development simplified, without being limited, thanks to the JSON information exchange protocol. In addition, the web platform has a set of instructions through text and video, making the process of learning and mastering the tool easier. Finally, the next step would be to allow the sending of commands from the browser to the device, thus being able to control some functions remotely as the load driving, relay triggering, among other functions.

## REFERENCES

ALLEN, G.; OWENS, M. **The definitive guide to SQLite.** New York: Apress, 2010.

ALVES, A. C. **Desenvolvimento de plugins**, 2016a. Available in: <https://bit.ly/2KPxSNA>. Accessed in: fev. 2019, in portuguese.

ALVES, A. C. **Wireless Monitor: aplicativo web para receber e mostrar dados vindos de equipamentos IoT**, 2016b. Available in: <https://bit.ly/2KEERtF>. Accessed in: fev. 2019, in portuguese.

ALVES, A. C. **Sensor de temperatura usando plataforma IoT Wireless Monitor**, 2016c. Available in: <https://bit.ly/2jpzoJR>. Accessed in: fev. 2019, in portuguese.

ALVES, A. C. **JavaScript SDK for Wireless Monitor**, 2017a. Available in: <https://bit.ly/2Nciiyg>. Accessed in: fev. 2019.

ALVES, A. C. **wm-example-esp8266 - Example using ESP8266 to send data to Wireless Monitor**, 2017b. Available in: <https://bit.ly/2ZbmZyH>. Accessed in: fev. 2019.

ALVES, A. C. **Demokit - app for building product demos and tutorials using web technologies with GNU/Linux support (Fork)**, 2017c. Available in: <https://bit.ly/2Z4IZMJ>. Accessed in: fev. 2019.

ALVES, A. C. ; JUCÁ, S. C. S. Wireless Monitor - aplicativo web livre para receber e mostrar dados

provenientes de equipamentos IoT. **III Escola Regional de Informática do Piauí. Livro Anais - Artigos e minicursos**, v. 1, n. 1, p. 17-22, 2017, in portuguese.

ASEMANI, M.; ABDOLLAHEI, F.; JABBARI, F. Understanding IoT platforms: towards a comprehensive definition and main characteristic description. 2019 5th INTERNATIONAL CONFERENCE ON WEB RESEARCH (ICWR), 2019, Tehran (Iran), **Proceedings...** 2019.

BANZI M.; SHILOH M. **Make**: getting started with Arduino. 3rd Ed. Make Media, 2014.

BEAN, M. **Laravel 5 essentials:** explore the fundamentals of Laravel, one of the most expressive and robust PHP frameworks. Packt, 2015.

BERGMANN, S. **PHPUnit - The PHP Testing Framework**, 2001. Available in: <https://phpunit.de/>. Accessed in: out. 2018.

BLANKENSHIP, G. **Laravel 5 Official Documentation**. 2019. Available in: <https://leanpub.com/laravel-5>. Accessed in: aug, 2019.

BROWSERSYNC. **Browsersync: time-saving synchronised browser testing**, 2017. Available in: <https://browsersync.io/>. Accessed in: mai. 2018.

CANTANHEDE, R. F.; SILVA, C. E. Uma proposta de sistema de IoT para monitoramento de ambiente hospitalar. VII Escola de Computação e suas Aplicações (EPOCA 2014), Santa Cruz (Brazil), **Proceedings...** p. 122–131, 2014, in portuguese.

CRUZ, V. S.; PETRUCELLI, E. E.; SOTTO, E. C. S. A linguegem JavaScript como alternativa para o desenvolvimento de aplicações multiplataforma. **Revista Interface Tecnológica,** v. 15, n. 2., p. 39-49, 2018, in portuguese.

GITBOOK. **Gitbook**: documentation made easy, 2017. Available in: <https://www.gitbook.com/>. Accessed in: jan. 2019.

GITHUB. GitHub help: about pull requests, 2017. Available in: <https://bit.ly/2gbWIFR>. Accessed in: jun. 2018.

GULP. **Gulp**: Automate and enhance your workflow, 2017. Available in: <http://gulpjs.com/>. Accessed in: aug. 2019.

HARDT, D. **The OAuth 2.0 authorization framework**, Request for Comments 6749**,** 2012. Available in: <https://tools.ietf.org/html/rfc6749>. Accessed in: jun. 2018.

HILL S. **Scalable IoT platforms**, 2019. 141 f. Master Thesis (Software Engineering)- Institute of Architecture of Application Systems, University of Stuttgart, Stuttgart (Germany), 2019. Available in: <https://elib.uni-stuttgart.de/handle/11682/10483 >. Accessed in: mai. 2019.

HUGHES-CROUCHER, T.; WILSON, M. **Node:** up and running: scalable server-side code with JavaScript. O'Reilly, 2012.

JONES, M.; BRADLEY, J.; SAKIMURA, N. **JSON Web Token (JWT),** Request for comments 7519, 2015. Available in: <https://tools.ietf.org/html/rfc7519>. Accessed in: aug. 2018.

JONES, M. B.; HARDT, D. **The OAuth 2.0 authorization framework**: bearer token usage, Request for Comments 6750, 2012. Available in: <https://tools.ietf.org/html/rfc6750>. Accessed in: mai. 2018.

JONSSON, J.; KALISKI, B. **Public-Key Cryptography Standards (PKCS) #1**: RSA cryptography specifications version 2.1, Request for Comments 3447, 2003. Available in: <https://tools.ietf.org/html/rfc3447>. Accessed in: jul. 2018.

JUCÁ, S.; CARVALHO, P. C. M.; PEREIRA, R. I. S. **Desenvolvimento de sistemas de Aquisição de dados sem fio**. Rio de Janeiro: Ciência Moderna, 2014, in portuguese.

JUCÁ, S.; PEREIRA, R. **Aplicações práticas de sistemas embarcados Linux utilizando Raspberry Pi**. Rio de Janeiro: Pod Editora, 2018, in portuguese.

KAMBOJ, P.; KRISHNA, C. R.; REDDY, S. R. N. Real-time implementation of scheduling policies for education using Raspberry Pi: a review. In: KRISHNA, C.; DUTTA, M.; KUMAR, R. (eds.) **Proceedings of 2nd International Conference on Communication, Computing and Networking. Lecture Notes in Networks and Systems**, v. 46. Singapore: Springer, 2018. p. 127-134.

KHARBOUCH, A. et al. Towards an IoT and Big Data analytics platform for the definition of diabetes telecare services. 2nd INTERNATIONAL CONFERENCE ON SMART APPLICATIONS AND DATA ANALYSIS FOR SMART CITIES (SADASC'18), 2018, Casablanca (Morocco), **Proceedings...**, 2018.

KRAWCZYK, H.; BELLARE, M; CANETTI, R. **HMAC**: Keyed-hashing for message authentication, Request for Comments, 2104, 1997. Available in: <https://tools.ietf.org/html/rfc2104>. Accessed in: mai. 2018.

LEACH, P.; MEALLING, M.; SALZ, R. A **Universally Unique IDentifier (UUID) URN namespace,** Request for Commands 4122, 2005. Available in: <https://tools.ietf.org/html/rfc4122>. Accessed in: jun. 2018.

LIU, J. et al. An effective biomedical data migration tool from resource description framework to JSON. **Database: The Journal of Biological Databases and Curation**, v. 2019, p. 1-9, 2019.

MILANE, A. **PostgreSQL**: guia do programador. São Paulo: Novatec, 2018, in portuguese.

JOYARD, N. **A fluent API to FFMPEG**, 2017. Available in: <https://bit.ly/2Zdalzo>. Accessed in: mai. 2018.

OLIVEIRA, I. F. **Desenvolvimento de um sistema de automação residencial baseado em IoT para controle e monitoramento de dispositivos elétricos**. 70 f. Bachelor Thesis- Universidade Federal de Ouro Preto (UFOP), Ouro Preto (Brazil), 2019, in portuguese.

POUWERKERK, P. **OS X screen recording library for Node (Fork)**, 2017. Available in: <https://github.com/pouwerkerk/screen-recorder>. Accessed in: mai. 2018.

PEYROTT, S. E. **JWT Handbook**, version 0.14.1, 2016. Available in: <https://bit.ly/2KMwKKA>. Accessed in: mai. 2018.

PORAMBAGE, P. et al. PAuthKey: a pervasive authentication protocol and key establishment scheme for wireless sensor networks in distributed IoT applications. **International Journal of Distributed Sensor Networks**, v. 10, n. 7, 2014.

QIU C. et al. Cloud computing assisted blockchain-enabled Internet of Things. **IEEE Transaction on Cloud Computing**, 2019.

RESCORLA, E. **HTTP over TLS**, Request for Comments 2818, 2000. Available in: <https://tools.ietf.org/html/rfc2818>. Accessed in: nov. 2018.

REVELL, S. **Internet of Things (IoT) and Machine to Machine Communications (M2M):** challenges and opportunities, 2013. IoT Special Interest Group. Available in: <http://bit.ly/2jeu4W5>. Accessed in: jun. 2018.

ROMERO, M. I. **PHP Authorization with JWT (JSON Web Tokens)**, 2015. Available in: <https://bit.ly/2k3lCud>. Accessed in: jun. 2018.

SEXTON, A.; ARO, E. **Messageformat**, 2017. Available in: <https://messageformat.github.io/messageformat/>. Accessed in: jun. 2018.

SHAH, J.; MISHRA, B. Customized IoT enabled wireless sensing and monitoring platform for smart buildings. Procedia Technology, v. 23, p. 256–263, 2016.

ROBOTJS. **The only Node.js first desktop automation library**, 2017. Available in: <http://robotjs.io/>. Accessed in: oct. 2018.

TEXAS INSTRUMENTS. **LM35 precision centigrade temperature sensors**, 2017. Available in: <http://www.ti.com/lit/ds/symlink/lm35.pdf>. Accessed in: jun. 2018.

TRAVISCI. **Travis CI:** building pull requests, 2017. Available in: <https://docs.travis-ci.com/user/pull-requests/>. Accessed in: nov. 2018.