A Web-Service to Monitor a Wireless Sensor Network

Rayanne M. C. Silveira[1(✉)], Francisco P. R. S. Alves[1], Allyx Fontaine[2], and Ewaldo E. C. Santana[3]

[1] Departament of Electrical Engineering, Federal University of Maranhão, Maranhão, Brazil
rayanne.silveira@aluno.ecp.ufma.br
[2] Université de Guyane, UMR Espace-Dev, Cayenne, France
[3] Departament of Mathematics, State University of Maranhão, Maranhão, Brazil

Abstract. In recent years, the interest in the Internet of Things has been growing, and WSN is a promising technology that could be applied in many situations. Regardless of the nature of the application, WSNs are often used for data acquisition, to obtain information from an environment of interest, so it is essential to consider how this data will be made available to users. Over the last years, an increasing number of web services have been used to deal with databases and final users, providing familiar interfaces and multi-platform access to those data. To address this problem, our study proposes a web application based on MVC architecture to monitor, organize and manage devices and data in a wireless sensor network. Here, is presented a functional evaluation of the proposed system and a discussion regarding the test results.

1 Introduction

Currently, the use of Wireless Sensor Networks (WSNs) has been a recurrent topic in several studies describing the different types of problems in which this technology can be applied. WSNs are communication networks composed of devices that exchange information through a channel. Usually, there are three types of devices: routers, end devices and a coordinator [1].

This scientific interest affects positively the popularization of this technology, since, coupled with the advances in microelectronics, there has been a growth in the development of commercial technologies that can be used for the construction of WSNs [2]. With the emergence of these technologies, the use of WSNs and, mobile sensing in general, has approached real problems, becoming a viable alternative to solve several problems [3] such as distributed monitoring, automation, and data acquisition, for example.

Regardless of their application, WSNs have their operation based on specific sensors to carry out the readings expected by the designer and to transmit them from wireless technology. These networks can capture a physical state from different points and concentrate this information in a single point for analysis.

J. Miller et al. (Eds.): ICWS 2019, LNCS 11512, pp. 126–146, 2019.
https://doi.org/10.1007/978-3-030-23499-7_9

After data concentration, a microcontroller is generally used to export the data in a way that they can be processed and used to generate useful information.

Tools that simplify the access and reading of the data obtained are essential in applications in which the sensors that make the acquisition are in hard-to-access locations. In precision farming, for example, a field that has invested in the use of technologies to obtain information that optimizes its production process, it is often unfeasible to collect the information only in the place where the reading was carried out, because of logistical reasons or even because of the impossibility of interaction with the place. Several methods have been proposed to help precision farming to gather data, and with the proper analytic tools, the changes during the season can be monitored [4].

Among the various tools that can be used to export the data obtained by WSN, web applications have shown to be promising tools. Furthermore, they allow the tracking of data collection in real time. Composed of a database and a graphical interface, this tool makes access to the desired data simple by providing access via a computer or a smartphone via the internet.

To address the problem of accessing information obtained in WSNs, in this paper, we propose a web-service based on MVC (Model-View-Controller) architecture to monitor, organize and manage devices and data in a wireless sensor network. Here, a functional evaluation of the proposed system will be presented and the test results will also be discussed. This paper has two main goals: (i) develop a web application to monitor and manage the values generated by the network and (ii) describe the interaction between the WSN developed for testing and the proposed system. The main contribution is the MVC Web service that was used to organize all data collected by a WSN, using a friendly and intuitive interface, and make these data available for many researchers and, as a result, increase the number of soil temperature and moisture databases. The next section presents a few works that have a similar proposal. Section 3 describes the proposed system architecture and functionalities. Section 4 shows the implementation of the system. Section 5 describes the experimentation and the results of the validation process before we present our conclusions.

2 Related Work

It is well known that the acquisition of data is a fundamental step for the monitoring of environments of interest. However, only performing the acquisition is not enough; it is also essential to allow the user to have easy access to this information. In the Internet, when it comes to the application of certain things, like Wireless Sensor Networks, there are many challenges such as energy consumption, routing protocols and, one of the biggest challenges: storing and interpretation of the data generated in those applications [5].

Something relevant is to make sure that the application, a WSN or another IOT application, has a low energy consumption. In [6] is presented an algorithm that reduces the energy consumption in heterogeneous cloud computing with a high success rate. When it comes to WSNs this type of proposal is prevalent, [7] and [8], for example, deal with techniques to save energy in WSNs.

In [5] the author presents a proposal similar to our paper, aiming to develop a web platform for data acquisition with tools for processing and analysis of data. The main objective was to develop a web platform, called CLUeFARM, which provides farmers with a way to monitor their farm from a distance through the internet. To assess the application, the author performed tests with response time for each service offered by the platform. Despite the satisfactory results, it is worth mentioning that the sensors and equipment used are complex and expensive. Our system uses a less expensive technology, the Xbee, that also has an easier configuration.

[9] presents a service related to soil management, within a platform that supplies integrated support for greenhouse activities, in addition to the CLUeFARM. Monitoring, assessing, and managing farm operations can provide enormous benefits in terms of productivity. To prioritize these items, the author proposes a service related to soil management, within a platform that supplies integrated support for greenhouse activities. In the work's experiments, a possible conclusion is that even though the proposed web-service is an extension of CLUeFARM, it may operate independently.

Another important topic was discussed in [10], in which the author presents a low-cost technology to build wireless sensor networks. The main gold was achieved, and the proposal presents a low-cost technology in comparison with technologies like the Libelium system. To deal with all the produced data, they also propose a Web Application. However, compared to the application of our paper, his application is quite simple, and just shows the values obtained with the WSN.

In [11], the authors propose a web-service to manage information from a green farm, like our project. The difference is that, in this case, the web application is a service-oriented platform. The main objective is to provide support in the management process and business development, and to increase the quality of products grown in farms. It also offers services for data processing and an algorithm to gather data. To validate their proposal, they made an experiment and an analysis to see the time needed to send notifications to users in case of an event.

In order to develop these services, a lot of frameworks and tools could be used. Knowing that there are diverse ways to build web applications, study [12] proposes a novel approach for Web service interface decomposition using a Formal Concept Analysis (FCA) framework. The authors made the validation of their method on 26 real-world Web services and obtained results that show that their method performs significantly better than other discussed techniques in terms of design quality improvement.

In [5,9,13], another type of framework is used: the MVC framework. Based in three layers of development, the MVC consists of models, views, and controllers. In each mentioned work, the authors used this kind of architecture because it promotes clean code, good practices, and it improves user interaction and interfaces. In the last few years, a lot of frameworks that use this architecture were used, the Laravel is one of the most popular ones.

Considering all presented platforms, our proposed system used a web-service as a tool to monitor, organize and visualize data collected by a WSN. To build this application, we have used the Laravel framework because it is a MVC framework, compatible with many browsers and it provides features that help in the fast development of scalable and modular applications [5]. Communications are based on HTTP protocol with the WSN to store the produced data.

3 Web-Service

Like any application, a web software is based on an architecture that will define how it will work. Our system is no different. In this section, we will describe the main features of our system: the architecture, the functionalities (system requirements and implementation) and the tools used to develop the application back-end and front-end.

3.1 Architecture of the System

The proposed system is a web service based on queries made by clients and answers sent by a server. The communication between client and server is performed using the Hypertext Transfer Protocol (HTTP), verbs (GET, POST, DELETE, and others), and it features a Representational State Transfer application (RESTful). Using this type of architecture decreases the number of queries because in this case all data will be sent using a single request into one JSON object through an HTTP POST request [14].

Usually, when the amount of exchanged information between an application and a web-service is big, data are converted into convenient structured formats such as XML [15] or JavaScript Object Notation (JSON). According to [16], in recent years developers have replaced XML with the JSON format to transfer data. Indeed, JSON is a lightweight data interchange format that is easy for humans to read and write, as well as easy for machines to parse and generate.

Looking for an efficient application with this behavior, our system was developed using a framework in PHP called Laravel. This framework is an open source tool for PHP development. It is largely applied as it has an organized code structure that makes it easier to read and modify the code. It also supplies tools needed for large and robust applications [17]. The Laravel uses a MVC architecture, therefore the development of the system followed this architecture.

The MVC is responsible for dividing the application in three layers of development: models, views and controllers. This design pattern is frequently used to develop web applications because it combines some technologies that are usually divided into a set of layers [18]. The model layer is responsible for organizing the data, communicating with the database and accessing all stored data. The view layer is the one that will build the graphical user interface. The controller layer is the most important and complex part: it is responsible for managing models and views, and processing the information from models to show in views [19].

The system works as described in Fig. 1. It shows the functionality of each layer and how these layers will interact with the final user. Our system will have a database to store all the data collected by the WSN. This exchange of information will be made by the coordinator device that acts like a client. It will send the information gathered to a cloud database through a POST request.

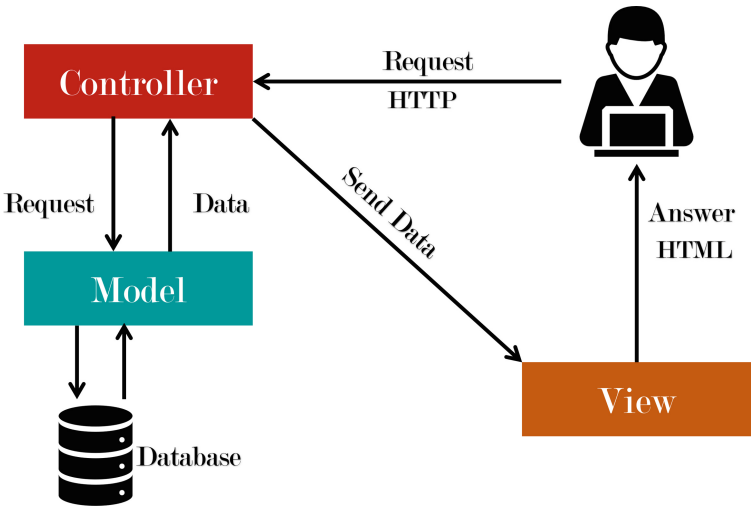


Fig. 1. MVC architecture

Therefore, the application must contain a functionality that gives the system the ability to organize in the database the data received by the microcontroller. The database model follows the entity-relationship (ER) pattern and is outlined in Fig. 2. There are three types of data: data, devices, and users, which are represented in the ER Diagram as entities.

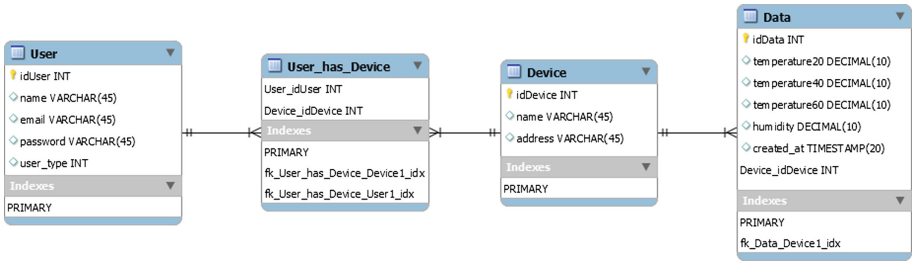


Fig. 2. Entity-Relationship diagram

The ER Diagram (Fig. 2) shows the relation between entities and their attributes. The Device entity has an id field, a device identifier (name) and

the MAC address of this device. Each device performs a series of readings, these readings are stored in another table, symbolized by the Data entity. This table contains fields for an id, for the values of the three temperatures, for the value of the humidity, data about the time the sample was sent to the system and, finally, an identifier of the equipment that has sent the value.

The User entity has fields that store information about users that can use the system. They have an id, name, email, password and can also fill a field to define which type of user they are. The last table in the database is the result of a relation between the user and the device. This relation was added in order to improve the functionality of a user and a device. The database is managed by MySQL, a robust and relational open source tool that is widely used in Web applications. This tool is already well-established in the literature, giving speed and reliability to the system [2].

Another important aspect for a Web service is its security. Our system contains a login system with password protocol security to restrict access to certain functionalities. The Laravel also uses the CSRF (Cross Site Request Forgery) tokens to prevent fake requests and has protection against XSS (Cross Site Scripting) and SQL Injection.

The web application will be hosted in a server and will be available on the internet. This requirement is important because this type of application was chosen, specifically, to use the WSN to collect data for future studies, like an international research project.

3.2 Functionalities

For the development of the system, we studied all the functional and non-functional requirements that the system should have. Functional requirements are requirements that express functions that a certain software must be able to perform or provide, while non-functional ones address system constraints and quality attributes [20].

Therefore, the functional requirements in Table 1 and the non-functional requirements in Table 2 were determined.

In Table 1, the first requirement (FR01) is the CRUD (Create, Read, Update and Delete) of users. This is not the most important feature of the system, but it ensures that new users can register to use, FR02, performs the same role, but for another entity, in this case it refers to the devices that make up the network. The third CRUD, FR03, performs the same role to the Entity Data.

Next requirements, FR04 and FR05, deal with the main functionality of the system: displaying the data obtained by WSN. The data display was implemented in two ways: through tables and graphs. Both data are filtered based on the address of the device that has sent the data. In addition to visualization, making data suitable for computational analysis is also important. Therefore, the requirement FR06 indicates the need for a function to export the data generated by the WSN in a format that simplifies their analysis. The format used was CSV (Comma-Separated Values), since it can be used to import data into several computational tools.

Table 1. Functional requirements

id	Name	Description
FR01	CRUD user	Create, read, update and delete users
FR02	CRUD device	Create, read, update and delete devices
FR03	CRUD data	Create, read, update and delete devices
FR04	Show data	Show in a table a data collection from a device in a defined time
FR05	Show charts	Generate charts with selected data
FR06	Export data	Generate and download a CVS with interest data
FR07	Link user and device	Allow the user to just see the data from a chosen device

Table 2. Non-functional requirements

id	Name	Description
NFR01	Uniqueness of users	
NFR02	Uniqueness of links	Between a user and a device
NFR03	Uniqueness of devices	
NFR04	Types of user	There are three types of users
NFR05	Admin privileges	Just an admin could create, update or delete
NFR06	Guest restrictions	The guest user can just read statistical information
NFR07	Common user restrictions	The common user can read statistical information and create charts, CSV files and make link with devices
NFR08	Create data	The data could just be created by WSN

The last requirement is the only one that has medium priority because although it is not an essential functionality of the system, it will contribute positively to its usability. This requirement (FR07) allows users to create a link with a certain device, so it can follow a device to which it is linked in a personalized page, facilitating the visualization of the data, since the network can have more than 100 devices.

In Table 2, some of the non-functional requirements are listed. The first three non-functional requirements are only system constraints, which should prevent conflicts and duplication in the database. NFR04 defines three types of users: admin, common and guest. The NF05 defines the privileges of an admin user. NF06 defines the limitations of a guest user. And, the NF07, defines limitations to the common users. The NFR08 is an important restriction, it defines that only the WSN can send information to the system.

After defining all the requirements, the functionalities were implemented, and the restriction and privileges were defined according to the non-functional requirements. To help the visualization and comprehension of the system, we present in Fig. 3 the Use Case Diagram that illustrates all the functionalities available for each type of user. In this diagram, it is possible to see the interaction between the WSN and the system.

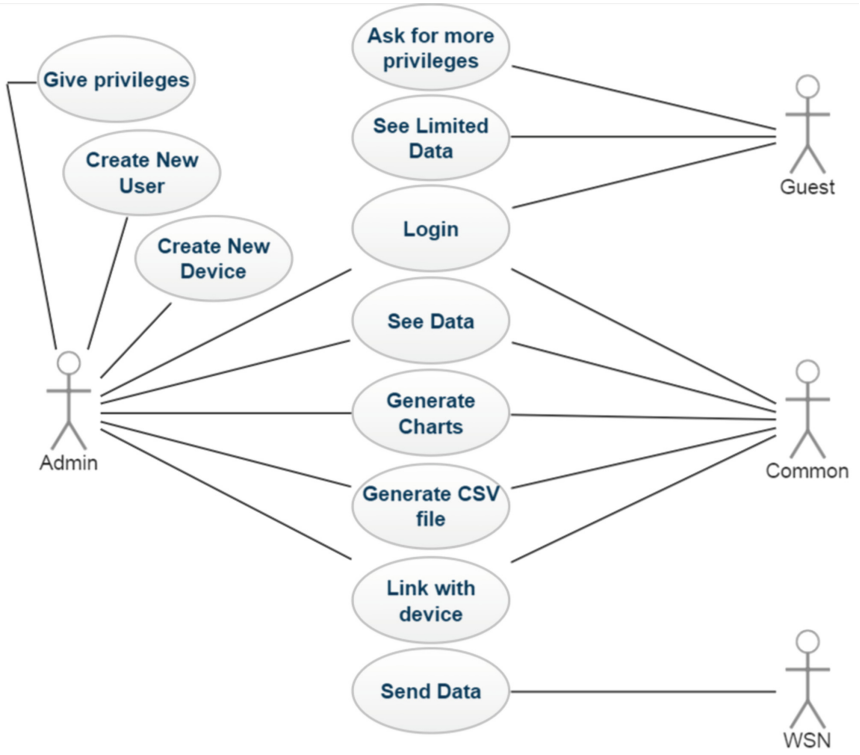


Fig. 3. Use case diagram

In this system, three main functionalities are highlighted and described below: see data, generate charts and generate CSV. All other functionalities play an administrative work that is very important. Besides the three mentioned, it is interesting to point out the importance of the "link with device". This function allows the user to select devices and to visualize more easily the desired data. Since our system will be used by researchers from different countries and the monitoring area for each device could not be related to everyone, this function is useful.

– **See Data:** the main objective of the proposed system is to monitor the data collected by the WSN, so it is fundamental to implement a function to allow the visualization of these data. This information is shown on a web page.

- **Generate Charts:** all data collected by the WSN will be used, processed and analyzed by a group of researchers. So in addition to showing data, it is interesting to show these data as temporal series, in charts, in order to produce more efficient analyses. These charts are generated according to the chosen device and could be personalized to show just information about one defined time interval.
- **Generate CSV:** this personalized application has a functionality that gives to the user the choice to generate a CSV file with all data collected by a selected device in a personalized time interval.

All these functionalities were implemented using the Laravel framework. This framework was used to develop the back-end and front-end of this application. As stated above, the system is organized in tree layers of development: model, view and control. In the next subsection we will describe the development and how these layers interact with the user and the system in general.

4 System Implementation

The development of our system consists in two layers: back-end and front-end. The back-end application is the place where logic is implemented. The front-end is where an intuitive interface is shown to the user; through this interface, they can access all the services listed in the back-end application [5]. To build these two layers we used the framework Laravel as a tool. As it was mentioned before, the architecture of the system is based on three layers of development: models, views and controllers.

- **Models:** three models have been developed in PHP using Laravel, one for each entity defined in the entity-relationship diagram (see Fig. 3): Data, Device, and User. Each model stores all necessary information to build an object for a given entity. A model will make the communication between the application and the database, and make all operations to read, create, update and delete information without using a query.
- **Controllers:** three controllers have been developed, one for each model. It was defined a DataController, a DeviceController and a UserController. The controller rules the application logic [13]. Each controller has functions that deal with the related data. For example, functions to create or delete a user are implemented in the UserController. A function is called according to the URL that the user puts on their browser. The control of that function will be called through a route made by a mechanism from Laravel, a route file that defines the patch and the corresponding function to execute.
- **Views:** it is the front-end of this application: all the web pages that compose the system. It is responsible for getting all information requests and showing them in a HTML file. There are many views in the proposed system, but just a few of them will be shown and discussed in this section.

The system has three perspectives: one for common logged users, one for guest users and one for logged administrator users (admin user). When a guest user access the platform and they are not logged, they only have access to a presentation page with some information about the project SenCSoil-Guyamazon, a Franco-Brazilian Program of cooperation in research, capacity building and innovation in the Amazon. From this page, the user can only access the login page (Fig. 4), where they can log in into the platform using an account or the register page and where they can ask to create a new account.

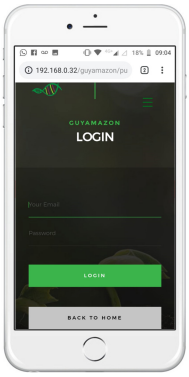
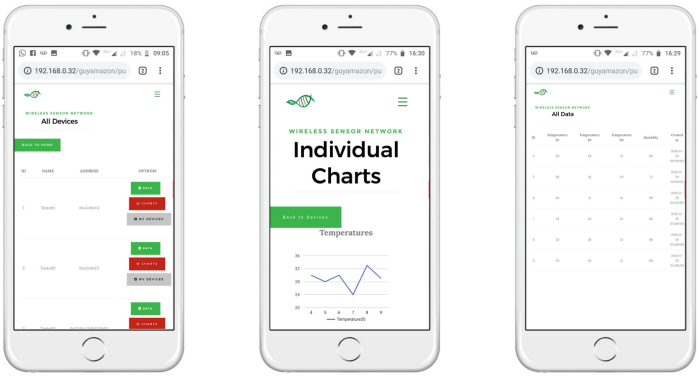


Fig. 4. Login page

After authentication, there is two kinds of users: the common and the admin. The main difference between those two is that only the admin will be able to access functionalities such as: accept a new user (giving privileges), create a device, create a user, delete devices or users, and update information in devices or users. The main functionalities of the system are available for both users. The first one, to see data, is accessed through the web page shown in Fig. 5a. As mentioned in the requirements subsection, there are two options to see the data, in a chart (see Fig. 5b) or in a table (see Fig. 5c). In the same screen, it is possible to make a link with a device, putting it in your "My Devices" list. The plot of the chart and table rows shows all data from the selected devices, independently of the time when it was created.

All devices for which the user has pressed the "add to my devices" option will be displayed on another page, illustrated in Figure 6. The large number of devices that the network can have, due to the great scalability of the system, make this mechanism essential to facilitate access to data and graphics of a device of interest.

The second main functionality is the chart generation, and the third, the generation of CSV files, they both have the same view pattern, illustrated in Fig. 7. The user must fill two date and time fields, one for the start date of interest and the second for the ending date. This time interval defines which data will be selected, thus seeking only those that were generated in the requested



(a) List of Devices View (b) Charts View (c) Table View

Fig. 5. Types of data visualization

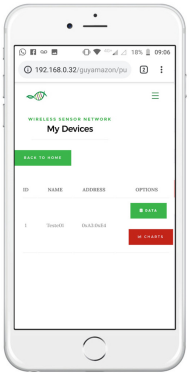


Fig. 6. My devices view

time interval. The other fields allow the user to select the device from which they want to export data, as well as the data type, since the network stores different types of data, such as temperature and humidity.

In the generate graph screen, after filling all the fields, the user is directed to a page with a chart, similar to Fig. 5b. If the choice is to generate a CSV, the file will be downloaded in the browser. In the administrator's view, in addition to the possibility to register users, the device display screen (see Fig. 5a) will also contain the options to edit or delete the device. The visualization in table format (Fig. 5c), to the admin user also has buttons to edit or delete the data from the system.

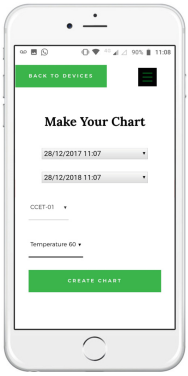


Fig. 7. Generate Chart/CSV view

5 Experiments and Results

This stage is divided into two parts. In the first one, we describe the development of the WSN and the algorithm that describes the communication between it and our system. In the second part, we make an evaluation of the usability of our system.

5.1 Wireless Sensor Network

To make some experiments and check the communication between the proposed system and the Wireless Sensor Network, we have used a simple network composed by XBee modules based on the 802.15.4/Zigbee protocol that built a low-power, low maintenance, and self-organizing WSN [21]. To build a WSN, after defining the technology used as a device (in our case the XBee), it is necessary to define the role of these devices in the network.

As mentioned previously, from a functional and operational point of view the nodes of a WSN can be configured as Coordinator, Router or End Device. According to [1], the coordinator is a device that manages all the functions that define the network in order to ensure its integrity and keep it active. In a ZigBee network, there is just one coordinator. The router is a device that can join an existing network, send, receive, and forward information to other network nodes. The end device is a device that can join existing networks, send and receive information, but cannot act as a messenger among other devices.

It is also necessary to define the quantity of devices and how they will be organized, more specifically the topology. The topology defines how the connections will be between all devices. There are three main kinds of topology: tree, mesh and star. In this paper, we have used the star topology because sometimes, mesh topology has a bigger number of lost packages than star topology [22] and this is an important measurement of quality of the service provided by a WSN. The star topology also has a lower delay rate compared to the mesh topology.

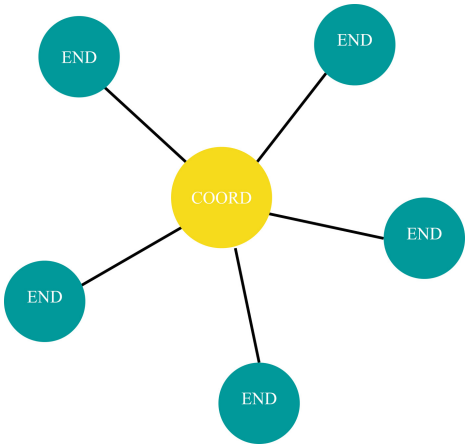


Fig. 8. Star topology

However, according to [8], in the case of precision agriculture, this is not a very important property.

In the star topology, as shown in Fig. 8, the coordinator node is at the center of the star and is connected to a circle of end devices. Any messages in the system must pass through the coordinator node which forwards them as required to devices in the network. The end devices do not communicate directly with each other and only communicate with the coordinator. The advantage of this topology is the simplicity of its implementation and the fact that the data packets between source and destination follow at most two communication links. The disadvantage of this topology is the lack of alternative paths, for the packets, between source and destination, and that all communications must pass through the coordinator.

Once the topology is defined, it is possible to implement the network. As mentioned before, to build the experimental WSN we have used the Xbee module, knowing that it is a practical tool that permits an easy and quick configuration of a network. The model used was XBee Pro S2C, powered by a 9V rechargeable battery. Since the input voltage of the XBee is 5V, an auxiliary circuit with a voltage regulator was used. The assembly made is shown in Fig. 9.

The first type of data to be collected by the proposed WSN will be of interest to researchers working with soil properties analysis. Therefore, the first experiments were carried out with the purpose of collecting different temperatures, from different soil depths and the humidity on its surface. For the acquisition of temperatures, the analog temperature sensor LM35, shown in Fig. 10a, was used and for humidity reading, the capacitive sensor shown in Fig. 10b was used.

In the XBee network, all the end devices send the data placed to the coordinator of the network, which is in a controlled location, with constant power. In addition to the XBees, the NodeMCU development module Esp8266, shown in Fig. 11, was used to perform the communication with the proposed system,

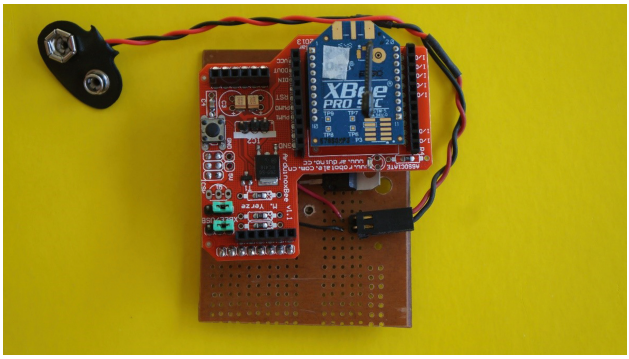
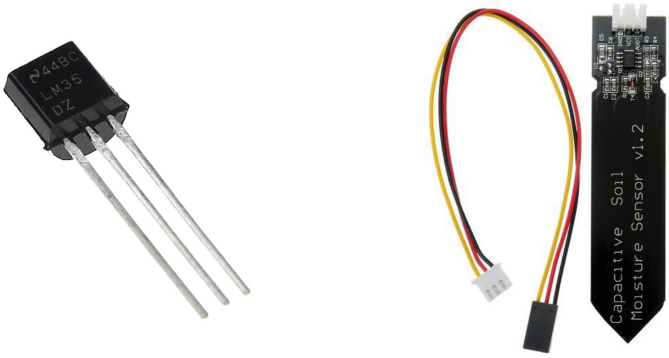


Fig. 9. Module XBee Pro S2C



(a) LM35 Temperature sensor (b) Capacitive humidity sensor

Fig. 10. Sensors used

which through serial communication with the XBee coordinator receives all readings from the network. In the NodeMCU, a routine has been implemented that interprets the data received from the XBee and formats it to send them to the database through an HTTP request. In this case, NodeMCU will play the role of Client and send the message in JSON format.

5.2 Algorithm in WSN

The software embedded (see Algorithm 1) in the NodeMCU was developed using the Arduino platform, in addition to the module's own Wi-Fi communication libraries, the JSON library, which performs the conversion of the message to be sent to the desired format. As mentioned, the connection between the NodeMCU and the Xbee Coordinator is made by serial communication. However, the data sent to the NodeMCU has a specific format, according to the communication

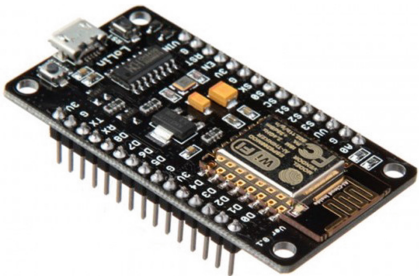


Fig. 11. Module NodeMCU ESP8266 [23]

protocol used by the coordinator. The XBee uses a predetermined frame pattern for each type of message one wants to send. Therefore, it was necessary to perform an interpretation of the frame before sending the actual values to the database. The frame has information such as device address, network address, configuration of digital and analog ports enabled, and lastly, the values read by each port.

Algorithm 1. Comunication Between WSN and Web-service

```
If there is something in Serial Port:
    Save Hexadecimal message;
    Convert To find real values of sample;
    Take temperature and humidity values and convert to JSON;
    Make connection with Host;
    Make a POST Request with the JSON message;
Else:
    Wait for a message;
```

The communication between WSN and the web application was successful. Therefore, to validate the interface, an usability analysis was performed. As the platform went through the evaluation stage, and it is not yet available online, we chose to perform an evaluation using heuristic and the cognitive walkthrough method.

5.3 Usability Evaluation

Heuristic evaluation is a method that identifies usability problems based on usability principles or usability heuristics, making it possible to evaluate the usability of the system [24]. According to [24], this kind of method is more efficient if it is combined with another evaluation method, like cognitive walkthrough. This one is a usability evaluation method in which one or more evaluators analyze a series of tasks, make questions from the user's perspective and check if the systems support the proposed goals.

To evaluate the proposed system, the method proposed by [24] was applied. Firstly, it is necessary to define all heuristics used in the process, showed in Table 3. After this is important to point out the scale that was used: (0) heuristic not applicable; (1) not fulfilled; (2) partially fulfilled; and, (3) fully fulfilled, following what was made in [24,25]. To make this evaluation we have selected three professionals with experience in software development and software engineering.

Table 3. Nielsen 10 heuristic principles [24,25]

id	Name	Description
HP1	Visibility of system status	The system should always keep users informed about what is going on, through appropriate feedback within a reasonable time
HP2	Match between system and the real word	The system should speak the users' language, with words, phrases and concepts familiar to the users. Make the information appear in a natural and logical order
HP3	User control freedom	"Emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo
HP4	Consistency and standards	Users should not have to wonder whether different Words, situations, or actions mean the same thing
HP5	Error prevention	Even better than good error messages is a careful design which prevents a problem from occurring in the first place
HP6	Recognition rather than call	Instructions for use of the system should be visible or easily retrievable whenever appropriate
HP7	Flexibility and efficiency of use	Allow users to tailor and speed up frequent actions
HP8	Aesthetic and minimalist design	Dialogues should not contain information which is irrelevant or rarely needed
HP9	Help users recognize, diagnose, and recover from error	Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution
HP10	Help and documentation	Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large

For the cognitive walkthrougth the author [24] proposes a task analysis and an interview with a questionnaire. For the interview, we use the four questions questionnaire proposed, and we added three questions from the Post-Study System Usability Questionnaire (PSSUQ) [26], an IBM questionnaire to evaluate usability of systems. All selected questions are listed in Table 4. The scale also

was applied to the heuristic: (0) not applicable; (1) not fulfilled; (2) partially fulfilled and (3) fully fulfilled. This evaluation was made by three selected developers that work with web applications development.

Table 4. Cognitive walkthrough questionnaire

id	Question
Q01	Is the control for the action visible?
Q02	Will the interface allow the user to produce the effect the action has?
Q03	Will users succeed in performing this action?
Q04	Will users notice that the correct action has been executed successfully?
Q05	Was it simple to use this system?
Q06	Was I able to efficiently complete the tasks and scenarios using this system?
Q07	Was the organization of information on the system screens clear?

5.4 Results

In the heuristic evaluation, it was possible to identify some problems in the interface, however it is important to point out that they are all linked to the absence of feedback in user operations, in which it was verified that it is not possible to identify success or failure in operations of addition or exclusion in the system. All the results are presented in Table 5, in which values from 0 to 3 determine how many valuers pointed to a given response.

Analyzing Table 5, it is possible to prove the previous assertion, since the items HP1, HP5, and HP9 indicated as not fulfilled are related to the absence of messages and the ability to retrieve the user error. The other problems encountered, HP6 and HP10 are related to the lack of help or system documentation that can contribute to the execution of the tasks by the user. In HP10, only one valuer considered this question incomplete, while one valuer considered it partially filled by the existence of a list of features and a use case diagram. Under the same criteria, the last valuer considered this information sufficient and considered it fully fulfilled.

Among the problems found in the majority can be considered of low gravity. However, it is worth mentioning that the absence of success or failure status in operations is considered as an absence of high severity, and that, due to the MVC architecture of the system, can be easily corrected. In a general analysis the evaluation was positive, with 50% of the heuristics at least partially fulfilled.

In the cognitive walkthrough evaluation, in addition to the answers of the questionnaire, comments were also made by the evaluators. First, two of the three evaluators believed that the functionality of editing a data produced by the network is unnecessary since the intention is to receive the values and that in a first analysis there would be no reason to modify them. Another problem

Table 5. Heuristic results

Heuristic	Not applicable	Not fulfilled	Partially fulfilled	Fully fulfilled
HP1	0	3	0	0
HP2	0	0	1	2
HP3	0	0	3	0
HP4	0	0	0	3
HP5	0	3	0	0
HP6	0	2	1	0
HP7	3	0	0	0
HP8	0	0	0	3
HP9	0	3	0	0
HP10	0	1	1	1

pointed out was a non-flexible database and web service, if the user wants to select a new sensor they have to change a lot of fields and adapt the views to deal with this new type of data.

In the analysis of the scenarios and the execution of the tasks, a few problems were found, the most relevant being the absence of confirmation messages of success or failure to create a device or user. In Table 6 are the answers of each valuer for the applied questionnaire. The id corresponds to the question, and 'Ev' represents each valuer.

Table 6. Cognitive walkthrough questionnaire answers

id	Ev 1	Ev 2	Ev 3
Q01	2	2	3
Q02	2	3	3
Q03	2	2	2
Q04	1	1	1
Q05	3	3	3
Q06	2	3	3
Q07	3	3	3

With the results presented in Table 6, it is possible to confirm that the scenario of confirmation of success in the operation was the most irregular in the view of the evaluators, just like in the heuristic evaluation. It is possible to perceive that, in a general analysis, the results of the evaluation were positive. The three questions from PSSUQ received a positive evaluation, even with the adaptation that was made, since this questionnaire consists of affirmations.

6 Conclusion and Future Work

The proposed system was developed to provide access to information collected by a Wireless Sensor Network, given that this is a tool that is becoming increasingly popular for data acquisition. In order to make information available in different countries and with an intuitive and user-friendly interface, a web-service has been proposed.

When it come to its development, a MVC architecture was chosen due to the organization and modularization of the code, facilitating possible changes, such as those pointed out in the evaluation stage. To confirm the communication between the WSN and the application, a network with XBee for the network composition and the NodeMCU device Esp8266 for communication with the web-service was used. The tests performed between the network and the application only sought to prove that the software developed for the embedded system could perform such a function.

The star topology used has limitations, once contact is lost with the coordinator this node is lost. However, the coordinator is in a controlled environment to prevent failures, and the distance between the nodes was not big enough to build a mesh network. In the future, to prevent failures in the coordinator, we can perform tests using a backup node that periodically checks to see if the coordinator has any problems.

The main goal of this work was not high security, a future work will be done in order to use new techniques to reinforce security. However, the analysis of the usability of the system allowed us to conclude that despite the negative aspects pointed out here, the application managed to reach the requirements for which it has been developed. This makes it necessary to execute new steps, firstly correcting the errors identified in the evaluation and, in sequence, putting the application online to analyze factor such as performance and other usability metrics with the final users.

References

1. Faludi, R.: Building Wireless Sensor Networks: with ZigBee, XBee, Arduino, and Processing. O'Reilly Media Inc., Newton (2010)
2. Wheeler, A.: Commercial applications of wireless sensor networks using ZigBee. IEEE Commun. Mag. **45**(4), 70–77 (2007)
3. Gai, K., Qiu, M.: Reinforcement learning-based content-centric services in mobile sensing. IEEE Netw. **32**(4), 34–39 (2018)
4. Huuskonen, J., Oksanen, T.: Soil sampling with drones and augmented reality in precision agriculture. Comput. Electron. Agric. **154**, 25–35 (2018)
5. Colezea, M., Musat, G., Pop, F., Negru, C., Dumitrascu, A., Mocanu, M.: Cluefarm: integrated web-service platform for smart farms. Comput. Electron. Agric. **154**, 134–154 (2018)
6. Gai, K., Qiu, M., Zhao, H.: Energy-aware task assignment for mobile cyber-enabled applications in heterogeneous cloud computing. J. Parallel Distrib. Comput. **111**, 126–135 (2018)

7. Xu, L., O'Hare, G.M., Collier, R.: A balanced energy-efficient multihop clustering scheme for wireless sensor networks. In: 2014 7th IFIP Wireless and Mobile Networking Conference (WMNC), pp. 1–8. IEEE (2014)
8. Rault, T., Bouabdallah, A., Challal, Y.: Energy efficiency in wireless sensor networks: a top-down survey. Comput. Netw. **67**, 104–122 (2014)
9. Serrouch, A., Mocanu, M., Pop, F.: Soil management services in CLUeFARM. In: 2015 14th International Symposium on Parallel and Distributed Computing (ISPDC), pp. 204–209. IEEE (2015)
10. Silva, M.S.d.: Rede de sensores sem fio de baixo custo para monitoramento ambiental. Master's thesis, Faculdade de Engenharia Elétrica e Computaçã - FEEC (2013)
11. Musat, G.A., et al.: Advanced services for efficient management of smart farms. J. Parallel Distrib. Comput. **116**, 3–17 (2018)
12. Daagi, M., Ouniy, A., Kessentini, M., Gammoudi, M.M., Bouktif, S.: Web service interface decomposition using formal concept analysis. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 172–179. IEEE (2017)
13. Gracia, J., Bayo, E.: An effective and user-friendly web application for the collaborative analysis of steel joints. Adv. Eng. Softw. **119**, 60–67 (2018)
14. Reynolds, D., Ball, J., Bauer, A., Griffiths, S., Zhou, J.: CropMonitor: a scalable open-source experiment management system for distributed plant phenotyping and IoT-based crop management. bioRxiv (2018)
15. Serrano, D., Stroulia, E., Lau, D., Ng, T.: Linked rest APIs: a middleware for semantic rest API integration. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 138–145. IEEE (2017)
16. Kao, K.C., Chieng, W.H., Jeng, S.L.: Design and development of an IoT-based web application for an intelligent remote SCADA system. In: IOP Conference Series: Materials Science and Engineering, vol. 323, pp. 012025. IOP Publishing (2018)
17. Wadkar, K., Koshti, P., Parab, D., Tamboli, S.: V-Buddy: a learning management system. In: 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), pp. 539–541. IEEE (2018)
18. Badurowicz, M.: MVC architectural pattern in mobile web applications. Actual Prob. Econ. **6**, 305–309 (2011)
19. Latief, M., Kandowangko, N., Yusuf, R.: Designing web database application for local medicinal plants of Gorontalo using MVC architecture. In: IOP Conference Series: Materials Science and Engineering, vol. 288, p. 012098. IOP Publishing (2018)
20. Sommerville, I., et al.: Software Engineering. Addison-wesley, Boston (2007)
21. Boonsawat, V., Ekchamanonta, J., Bumrungkhet, K., Kittipiyakul, S.: XBee wireless sensor networks for temperature monitoring. In: the Second Conference on Application Research and Development, ECTI-CARD 2010, Chon Buri, Thailand. Citeseer (2010)
22. Soijoyo, S., Ashari, A.: Analysis of Zigbee data transmission on wireless sensor network topology. Analysis **8**(9), 145–151 (2017)
23. Kodali, R.K., Soratkal, S.: MQTT based home automation system using ESP8266. In: 2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), pp. 1–5. IEEE (2016)
24. Pilco, H., et al.: Analysis and improvement of the usability of a tele-rehabilitation platform for hip surgery patients. In: Nunes, I.L. (ed.) AHFE 2018. AISC, vol. 781, pp. 197–209. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-94334-3_21

25. Nielsen, J., Molich, R.: Heuristic evaluation of user interfaces. In: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 249–256. ACM (1990)
26. Rosa, A.F., Martins, A.I., Costa, V., Queirós, A., Silva, A., Rocha, N.P.: European Portuguese validation of the post-study system usability questionnaire (PSSUQ). In: 2015 10th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1–5. IEEE (2015)