Secure management and integration system for electrical devices

Bîrsoan Daniel Florin
Technical University of Cluj Napoca
G. Baritiu 26-28
birsoanf@gmail.com

Ștefănuț Teodor
Technical University of Cluj Napoca
G. Baritiu 26-28
teodor.stefanut@cs.utcluj.ro

DOI: 10.37789/rochi.2020.1.1.13

ABSTRACT
The main purpose of an Internet of Things(IoT) network is to make people's work and life easier by providing processes and services as close as possible to their needs. Globally, it is stated that the Internet of Things (IoT) must be available everywhere. As the Internet is almost ubiquitous today, this is not an unreasonable requirement. But to create such a network, it would be necessary for all devices, regardless of the date of creation or manufacturer to be able to be inter-connected to a common platform and made accessible securely through the Internet.

In the current article we are proposing an architecture that responds to this need for the interconnectivity of devices and facilitates secure communication through its components. Through the installation of dedicated board/boards in the desired space and through the connection of the electrical devices on different pins to them, the “objects” are connected to the internet and the user can control their ON/OFF state remotely. So this purpose the proposed architecture features four main components: (1) on-site boards that control the electrical items and the internet connection; (2) a server that orchestrates the communication between all the other components; (3) a web application for electrical items management; (4) a smartwatch application for electrical items control.

Author Keywords
System Security; Modular Software System; Architecture in Software Application, Web Application, Smartwatch Application;

ACM Classification Keywords
•Hardware~Communication hardware, interfaces and storage~Sensors and actuators
•Security and privacy~Software and application security~Web application security
•Social and professional topics~Computing / technology policy~Privacy policies

INTRODUCTION
The Internet of Things (IoT) was first mentioned in 1999 by Kevin Ashton. A common vision in this area is that in the future will be a single global IoT communication network, because the amount of information become huge and there are more and more devices. And another fact is the reliance on the word "things" referring to all the physical objects that surround us.

IoT is an area that have developed a lot in recent years. It is this evolution that has brought more and more IoT solutions develop to the market that has created customized products and aimed at attracting end customers in their ecosystem. Following this diversity, it has not been possible to operate on the initial architecture model for many years and many architectures and ways of working have emerged to manage the very large volume of data and very heavy network traffic.

Also, the interaction and the way we relate to this area must be very well defined and easy to understand by everyone, because, for even greater popularity, all people regardless of their technical knowledge must be able to use the solutions simply and effectively in order to meet their needs. That is why more and more investors in the smart devices development market are migrating to minimal user interactions with the system or systems that manage their space through various innovations such as applications for virtual assistants, Alexa or Google Assistant, in various smart objects, applications for smartwatches or smart TV applications. We must not forget the fact that often the one that facilitates the access to the devices attached by the user in a system are the hubs. As they become more accessible, they have taken over much of the market for the simple needs of users, especially those with built-in assistants, which also help stand-alone applications to communicate with devices.

Following the minimalist interaction described above, I must highlight the contribution of smartwatches in this direction and their rapid evolution in recent years. The clock, after all, over the years, from its appearance to the present day, has been worn by richer people and people representing the middle or lower class. Everyone has become accustomed to his presence, his behavior, and his usefulness in giving us the exact time. It is this habit that has helped the further development of the field of smartwatches. The first smartwatch[1] appeared in 1972, produced by Hamilton Watch and Electro / Data Inc. which was just a digital representation of time in the form of Arabic numerals. Later, it reached an industry that sells

[1]http://www.mobileindustryreview.com/2016/10/33860.htm

over 2.1 million devices annually, most of which are Apple devices.

The smartwatch has a great advantage in managing personal smart devices compared to the regular phone because it is carried on the hand and often replaces its functionality by 80% so users who have such a device are more tempted and satisfied with applications for managing smart objects on it. Its minimalist design, small screen, limited processor and RAM are so far the biggest disadvantage that blocks it from completely replacing the phone.

Security, which is one of the fundamental problems of IoT systems, has been ensured in the proposed solution through the encryption of all communication using tokens. Usability aspects have been addressed through the development of minimalist and easy-to-understand client applications.

RELATED WORK

The book [4] begins with a consideration that specifies that the IoT domain is becoming as abstract as the "Big Data" domain, and how we relate to it must become increasingly personalized. Like development solutions to problems in this area, in this domain we can no longer operate on the "one size suit all" model for years, and this is described in a very objective manner in the chapter [2].

Studies show that sooner or later a single dedicated IoT network will be needed [4] and that all objects will communicate through it. Of course, converting to IPv6 will be a big step forward in this endeavor because the number of public IPs would increase exponentially [1] and every device or hub in space would have one.

And security is one of the most important aspects of the field. The growing number of devices and their holders requires the encryption of sensitive end-user data. Depending on the type of attack, like man-in-the-middle attack or false node message corruption, both the data sending device and the node/server that manages it must be prevented from stopping communication. A list of such attacks can be found in chapter [6], which also describes possible implementations of solutions for each main attack being encryption, object authentication, Datagram Transport Layer Security, or Information Flow Control.

One of the long-term success criteria of a system is the architectural type chosen to develop it. The big developers in the market for object management services in a smart way do not reveal the whole architecture on levels but only large explanatory diagrams or small portions of text that result in how to do things.

An example compared to the system described in this paper is openHAB, which is a company developing custom IoT solutions. In both systems there is the concept of modularity and decoupling of logic data sources. Another existing solution on the market with which the developed system is similar is the application from Samsung, ie SmartThing. From the structure information provided by the developer on the official page of the application we can learn that it relies heavily on the integration of devices in an external server from where a system kernel provides access to applications for customers. As a communication architecture it would be assumed that they use the Client-Server type, similar to the system described in this paper, because they have endpoints through which data is extracted and they must be called by an application or a third party to provide data or perform tasks on the server.

So the competition is given by the diversity of IoT products and applications/systems. Applications such as SmartThing, openHAB, or Google Home, which have gained a lot of ground in recent years due to their scalability and availability, are the main competitors of the developed solution. The competition, after a careful analysis of the market, is based on IoT devices that have either wireless or Bluetooth in their management and integration as opposed to the system designed by us where devices without these two features can be integrated provided they are connected to a power source and operate on the ON / OFF principle.

Solution	Security	Applications dedicated to the system	Electrical consumption of devices
SmartThing	Yes	Mobile/Web/SmartWatch	No
GoogleHome	Yes	Mobile/Web	Yes
openHab	Yes	Mobile/Desktop	Yes
AFHA	Yes	Web/SmartWatch	Yes

Table 1: Comparison between the current solution and competition on the market

Thus we offer the possibility to automate the spaces without assuming the expenses related to the purchase of new, more expensive devices, which are compatible with a certain system, by integrating our system in the space and connecting the existing objects to it.

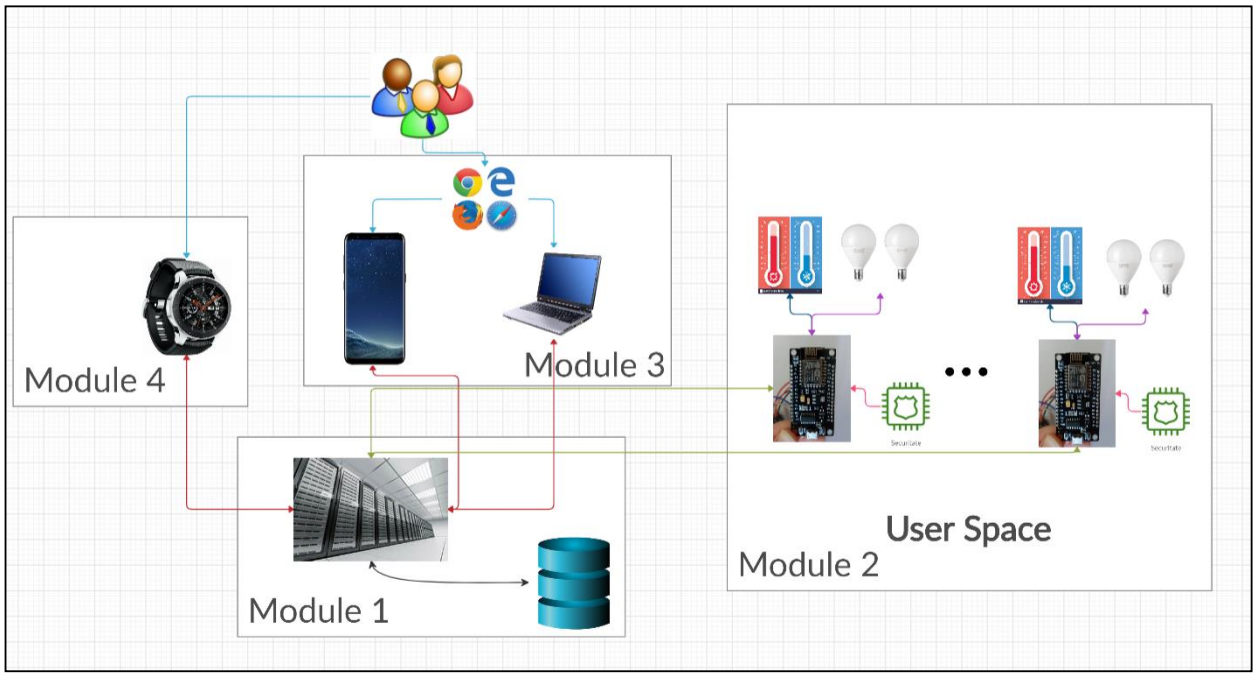

Figure 1: Architecture design.

IMPLEMENTATION

The system consists of four main modules that have been divided according to the functionalities they perform.

The first module, and the center of the system, is the server-side that appears at the bottom of Figure **1**. It is responsible for interconnecting the other components and also responsible for generating the authentication keys for the system. Also in this module is added the database where all the data from the applications are persistently maintained, from the authentication data, where the password and the pin are encrypted with a sha256, to the sensors and devices data from spaces attached to the system.

The second module is the hardware part of the users' buildings. On the diagram, it is on the right side and the plate is symbolically represented by a mini-hub and the bulbs and thermometers represent the rest of the connected devices. The module has the responsibility to connect the electrical devices to the central server through a board that serves as a hub. At the same time, it constantly requests data at regular intervals from the server to find out if it is necessary to execute commands on the devices. The board has an authentication system that request a token from the server before sending the user data and all requests, after this point, are accompanied by the key received after authentication. It also at a certain interval transmits data from the person's spaces to the server to maintain the consumption history and the history of biometric data.

The third module is the web application that has been designed in a way that is compatible with multiple screen sizes. This module is responsible for managing action groups that the user can create for their own spaces, with the ability to activate one or more devices at once. It will also display tables with adjustable consumption, temperature, and humidity depending on the chosen building or the desired time interval. On the architecture diagram, it is represented by the phone and the laptop and through the search engines, the developed application will be accessed. In this case, too, a prior authentication will be made and a key will be received and then used in all requests made to the system. This application is designed to be scalable on both mobile and a desktop screen so that it can be accessed by all potential customers regardless of phone brand or of course whether or not it has a computer or laptop.

The fourth module is responsible for activating the action groups defined in the web application. For the implementation of this module, the development of an application dedicated to smartwatches was chosen because the aim was to control the attached components as easily and quickly as possible. The application also benefits from a notification system through which the end-user is notified when the temperature or humidity exceeds a normal threshold in his home to prevent any incident in the person's premises. Notifications are also received when the application runs behind the others and if more come in a short time they are merged by the system into a larger

notification where they are displayed in the order in which they came to the smartwatch. Given the increase in quality in the field of these smart accessories, the implementation of the solution was much more affordable, but still, in terms of hardware specifications, smartwatches lag behind phones and do not provide a complete transition from phones to them soon. This module is represented by the Samsung smartwatch on the left side of the diagram. The communication between each module and the main server is made using the internet networks and the http1.1 format and a key generated following an authentication made to the server is attached to it. Also, to make communication more efficient and to facilitate the openness of the system as a functional requirement, a common language based on the JSON data structure was implemented. Even if one of the components disconnects from the system, it will still be able to operate based on these formats.

The diagram also describes how users interact with the system. It is observed that they have access to the smartwatch application and the web application through browser on the phone or laptop or computer.

Server

The need for a central server is given by the control of the large data flow that the attached components manage, leaving them to focus on managing their main functionalities. It links the persistent part with the user data and the modules to which it provides processed or raw information depending on the need of the place where they are requested. For the easy integration with the rest of the modules and the use, without explicit direct intervention from the programmer, of several threads to perform the tasks, a server designed with Java was used in an application such as Spring Boot.

The server is created in a Layered architectural style with 3 separate levels for the database part, for service logic, and for the connection to the outside through web service.

Security

The security part is provided by the server by generating a token after successful authentication. Once the modules have made this registration and have the token they attach it to the header part of the http1.1 in each request they make to the server. The token part was implemented using the io.jsonwebtoken library from which with the help of an ally generated key for an encryption algorithm a string is obtained through which the personal information about the user who made the request is hidden by the attacker. The algorithm used to generate a token is HMAC which generates a hash function after a private key, for creating the message in the hash sha512 is used for a longer key length and for increased security.

However, knowing the problems that could arise related to the encryption algorithm and the fact that it could be broken and the attacker could take over the secret information entered in the token, we added another level of encryption for the user's data. Before adding the information for encryption, the text is transformed into an integer, the integer is corresponding to the character values in the ASCII table. To secure the text we also used a vector of characters with a randomly chosen size between 20 and 40, the range was chosen arbitrarily to generate a key large enough. Then scroll through the text transformed into ASCII and if the corresponding number in the table is below 9 add two zeros before the number, if the number is less than 100 only add a zero before it and if it is greater than 100 it will not be added any string. After the newly created value, an extension represented by the above fixed-length vector, called the jump, with random values, will be added. The next number in the original string will again generate a vector with the same size but different values and will be added to the new string in the original text. All steps will be repeated until all text is consumed. And finally, we will move on to the classic encryption of the new format string.

To maintain a constant key to generate the token and the size was chosen for the string that is added after the ASCII values, we used the Singleton design pattern in which the class responsible for generating, validating and decrypting the token messages is instantiated once and the data from class is reused.

However, in order not to risk reusing the older authentication keys, the key and the vector size will change every 12 hours with a cron job set on the Spring Boot application. This was only possible by allowing the spring application to be scheduled in the main class with a specific annotation.

And for the decryption part of the message will be considered the first 3 digits of the final text as the first letter with the corresponding value in the ASCII table, after knowing the size of the string of numbers added after it will be iterated over that number of values to the next value of 3 digits which is the second corresponding letter in the ASCII table. Repeat the steps described above until the size of the final message is reached and the original text is obtained. Pre-encryption of the original message brings a security bonus assuming that the attacker could decrypt the hash function used in the token.

Therefore, on two successive calls to generate a token for use in system requests by a user named "vas.ile1", it will look like this for the first time.


```
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIxMTg2MzAxNjczODE1NDcwMjIzNjE2NTUwMTM2MT
A5NzQyNDIxNDUwMDU1NzE3NDE0MDE4MzE0NzA2MTA4MzMzMjY4MTM4NzQzNjE3NDI4ODgzM
DY3MDMwNDY3Njg0NDAxNDQ3NDU2MzcwNDA1MTExMTQyMzEwNTY3MjcyNTM4Njc4NTQ2MjMy
ODI0MzU0NjAxMTA4MDU1NTg1NzU4NDIzNjg4NDUyMzQ4MzY3NzYxMDE3NjgyMDYxMDU2NDc
4MzQyMDc3ODQ4MjI0NDA0OTI0NjE1ODEwMDIxNDY2MTM3MTczMzAyMTQ3In0.x9p78lRRUE
JB_NkdOA8DjOx_JAHVP2m-8suY8XawqACO7-r8XOxEN7Gosml_mWp3-0eXX1MiSRJ21aR78
_HJ-g
```


Figure 3. The first authentication key for the name vas.ile1.

And for a second request, the authentication key to requests to the server will look like this, different from the first authentication.

eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIxMTg2MTQ3MDczMTMxNDAwMjY4MDQzMzU0MTExNT
A5NzA2MjAxNjUxMTYxMjYzODU3NDEzMDc0MzA0MTA4NDcyNzQ3MzMwMDg2ODgyNjYwNTA2M
jg0MTQwNDYxODYzMTY1NTU4ODU1NjA0NTEwNDgxNTIwMTEwNTI0NDIyMjcwODIyNjg0NjQy
Nzc2Mzg0MzM3MTA4NDQwNTM1MTc3NzQ0NzUzMTE4MDU4MDY1MTAxMDEwMDA0MzI1NTc1MDM
3MjY3ODU4MDYxODU0NTA0OTUyNDcwODAzMjIxNDYyMzU0NDMxNTc3NjQwIn0.i29njL9e02
q6ADgggh0Ou0xBRx5JmpO2FAV-1GjiZJnauhOCvEd2renBWSzw48x_LcGoR6mlrvulK97Tx
5Y8FQ

Figure 4. The second authentication key for the name vas.ile1.

It is also worth mentioning that during the 12 hours until the renewal of the key for generating token encryption functions both generated examples remain valid and the system will treat them equally for the user who owns them.

Hardware integration

A NodeMcu v3 - Lolin development board containing an ESP8266 chip was used to integrate the devices and sensors. The chip facilitates the connection of the board to an internet network thus creating a connection between the main server and it.

A client-server architecture was used to communicate and exchange information between the development board and the server over the Internet, to the detriment of a publish-subscribe architecture. We went on this so as not to make communication very difficult and not to block certain channels as the second one described did. If we went to the second one a port on the server side was continuously busy to detect certain events and thus the system became limited by the number of ports available on the machine where the server application was running.

In the context of the client-server architecture, two modes of communication with the server module were considered, the one in which the server searches for the board to query and transmit information and in which the development board searches for the server to receive information about the tasks to be performed and sends him the latest sensor readings via http1.1 messages. At this point, we went for the second option to disconnect the server from the boards and not need changes on the server-side when adding new spaces and buildings to a person.

The data is sent in JSON format using the ArduinoJson library created by Blanchon B. which provides support for data serialization in chapter [2] and for their deserialization in chapter [3]. Also in this way, after authentication, the authentication key (token) is taken over and stored in a character vector on the board, being used later in the requests made to the server. The size of the vector with which the data is unearthed from JSON has a fixed length of 800 characters because performance reasons the dynamic allocation vector for the library used to extract the data was removed.

For the reading part of the sensors, the data is taken from a DHT22 sensor that provides temperature and humidity in the form of float variables and then with them a JSON is created and the information is sent.

Each device connected to the server has a unique identifier for the board that is attached to it for a specific space. In the implementation part related to the switching on or off of the connected devices, a list is sent from the server with these identifiers in the form of a string for example “012”. The list is deserialized and all the devices that are in the list are turned on and the rest remain off. The list is built according to the needs and settings of the client. The limitation of the hardware system here was to 3 devices but the aim is to add a much larger number of devices to the development possibilities.

Client applications

On the client-side, there are two applications which focus on different functionalities. The first one, implemented as a responsive web application, is focused on managing and viewing device statistics on a computer, laptop, or mobile phone. The other, allows the user to control the devices, more precisely their activation, and also to receive notifications in case of exceeding certain normal values for the owned spaces. The Tizen system has been used for the development of this second application, ensuring compatibility with smartwatches that use this system.

The web application is designed to be scalable on both the desktop and the phone. Also, the web application is built in a high usability mode with big buttons and also written big enough not to make it difficult for the user to perform the tasks.

The security for unauthorized access in web application is primarily held by the index class where page routing is based on URLs. When a user does not have an authentication key set in the local storage part of the browser, they are not allowed to access the page and are redirected from the router to the unauthorized page. This is done by a URL analyzer that looks at how the URL is formed and if it recognizes it, in case of “/” it goes to the authentication page, or in case of “/ client” it checks the authentication key from the local storage and if it exists enter the page. If the key exists and is not valid, the user enters the page, but does not see any data because at the time of requests to the server it rejects them.

Also, on the web security side, XSS type attacks were taken into account, which allow the attacker to run a program inside web pages through which he can extract data about the client and about the traffic on that page. This attack is not limited to inserting code by common means such as fields where data is inserted directly but can also be inserted into tables or menus with multiple selection. For this reason, in each field where data can be inserted, they are considered text by the application. On the tables side, the code for them is generated dynamically so that the attacker

cannot insert static code in one of the table options. The same principle applies as in the case of tables with multiple selections. On the design side of all the visual components described above, it was ensured that no line of code that could be run or selected was in the control of the browser, thus blocking access to the application code in a direct way.

The smartwatch app is compatible to run on Tizen 4.0 and higher. It uses a readable menu that contains buttons that have the person's buildings on the first level, on the second level after clicking on a building you can see the person's spaces to the building and on the last level are user-defined action groups attached to that space. The user-defined action groups, with the devices you want to start when activated, are red if the scenario is inactive and the devices are off and green if the devices are on and the scenario is active. When the user presses a green scenario, all devices close and the scenario turns red.

For the authentication part, a minimalist design was created with few elements in order not to visually load the user and to make the interaction with the system as easy as possible. The distance between the text where the information to be entered is specified and the space allocated for input is left for a hidden element where the text specific errors are entered when it occurs. The error display mode and the authentication screen are visible in Figure 5.14. Also, a detail of the design part is that all the elements are centered and expand according to the space available on the clock screen. The application has been designed to have the same design experience on multiple watch sizes and sizes. The menu was stacked with centered and dynamically allocated elements when creating the page. If the elements take up more space than available on the device running the application then a right-hand browser will be autogenerated that will allow them to pass through, such an example can be seen in Figure 5.15 in the clock on the left. Also, each menu has a button fixed at the bottom that allows the user to return to an internal action in case it is wrong, but this action does not affect the values that are already saved that have been selected by the user, ie not remove. But a new press of a menu item and the default move to the next level involves overwriting the values of the user's actions.

Also the smartwatch application will receive notifications about the high temperature or high humidity from the server upon request and will display them to the customer on the watch.

For the notifications part, a multi-threaded solution was used. The application runs on a certain thread and before making the first load, from a cycle in which it remains on, it creates a thread on which checks are started in connection with the customer alert situations. The thread, which contains the task of checking notifications, has a timer that is set to 10 seconds, so it queries if new information has appeared about the status of the client's spaces. If the list is not empty, the resulting text is transformed into a list of notifications and sent to a service that sends them to the customer.

CONCLUSION

Pursuing the level of security as one of the requirements of increasing interest in IoT devices, I focused on this by providing token-based communication in all modules and the project as the first version achieved all its objectives.

It can integrate electrical devices that work on the principle of ON / OFF, provides secure communication between its components, and have also been created in a modular style aiming to decouple the components that make it up and a minimum dependence, only in terms of data format. An additional security level has also been implemented for the token generation by encrypting user data with an algorithm that takes into account the ASCII codes of the characters.

Client applications also provide security when communicating data with the server and are highly user-friendly, simple, and easy to use without the need to perform many steps to perform the desired tasks.

REFERENCES

1) Al-Anqoudi Y. S., „Internet of Things," 1 February 2020. [Interactiv]. Available: https://www.researchgate.net/publication/339383844_Internet_of_Things. [Acces 14 March 2020].

2) Blanchon B., „Serialization tutorial," [Interactiv]. Available: https://arduinojson.org/v6/doc/serialization/. [Accessed 23 January 2020].

3) Blanchon B., „Serialization tutorial," [Interactiv]. Available: https://arduinojson.org/v6/doc/serialization/. [Accessed 23 January 2020].

4) Croes E., Software Architectural Styles in the Internet of Things, Nijmegen: RADBOUD UNIVERSITY NIJMEGEN, 2015.

5) Hassan Q. F., Internet of Things A to Z: Technologies and Applications, Al Manşūrah, Egypt: Wiley-IEEE Press, 2018.

6) Jurcut Anca D., Pasika S. Ranaweera, Lina Xu, „Introduction to IoT Security," in IoT Security, Dublin, John Wiley Sons Ltd. 2019, 2019.

7) Nayak P., „Internet of Things Services, Applications, Issues, and Challanges," in IoT, Hyderabad, India, Gokaraju Rangaraju Institute of Engineering & Technology, 2019, pp. 354-366.

8) Santhakumar R., Subramanian B., „IoT Technology, Applications and Challenges: A Contemporary Survey," Wireless Personal Communications, p. 27, 10 April