

# Implementation of Interface for Device and Smart IoT Platform Integration

Francisco Javier Flores Zermeño<sup>1</sup> and Edgar Gonzalo Cossio Franco<sup>2</sup>

<sup>1, 2</sup> Posgrado CIATEQ A.C., Centro de Tecnología Avanzada,  
Mexico

franc.javier.flores@gmail.com, kofrran@gmail.com

**Abstract.** The implementation described in this research aims to show the parts that make up an intelligent IoT system. The system is adaptable to several technological applications that require sending information to the cloud. The sending is done by using devices or hardware that have the resources to connect to the Internet by applying the HTTPS protocol (Hypertext Transfer Protocol Secure) for sending and receiving data. An API (Application Programming Interface) is implemented to allow the exchange of information between the device and the server. The logic applied in this API was solved in a simple but not limited way where its main feature is to have authentication for IoT devices, besides supporting only post requests, excluding unauthenticated devices. The responses from the server to the IoT device use the HTTP protocol (Hyper-text Transfer Protocol), which allows the use of statuses for each response. The responses can be of different types, such as errors in the request and successful transactions that return the requested results. In such transactions, additional logic facilitates reprogramming the IoT device's behavior. For example, it is possible to modify the reporting times, reboot the device, update the credentials for periodic authentication, even enable and disable the sensors that integrate the IoT device, among other possible uses not yet explored.

**Keywords:** IoT, Intelligent System, Intelligent Device, API, Telemetry.

## 1 Introduction

Currently, applications based on the Internet of Things (IoT) are created to be used in different devices and web services in order to improve processes from manual to automated. With the above, a wide variety of problems in smart cities and industry automation are solved [1].

Devices developed or adapted for use in the Internet of Things (IoT) can communicate in different ways, using a variety of protocols to connect to the Internet. It is worth mentioning that there is no single best protocol that can cover all solutions, but it is possible to choose the most suitable ones, depending on the specific needs of each application. Cloud devices and services can be represented as reusable

components to be adapted in multiple ways. Internally such devices contain complex logic that is developed with low to high level object oriented code, as well as protocols and interfaces that are fundamental in the integration to create adequate workflows [2]. Integrations between IoT oriented intelligent systems involve the fusion of hardware, software, server like operating systems and their VPS (Virtual Private Server) virtualization [4]. This IoT design pattern integrates several layers ranging from the feature layer of the IoT device to the means by which telemetry is sent and the interface or data exchange through the API [5]. The three points mentioned above are described below:

- IoT device layer: its main function is to detect environmental variables, they are devices that execute instructions through the use of actuators and collect information through sensors. These devices analyze and discriminate the information to leave only the relevant data and with them build and prepare the telemetry, which is sent to the IoT platforms.
- Internet layer: interconnects the devices with each other and enables the exchange of information; these are systems that manage the acceptance and transmission of IoT data.
- Interface layer: commonly used to facilitate the connection between devices, it is also used for connections with API or data transfer protocols to support the needs of applications, in addition to processing the data to convert them into useful information and present them through a visual interface for users on IoT platforms [3].

An API is a set of rules that determine how the service can be consumed based on the parameters set for the connection, as well as the responses returned by the connection [14]. There is some intelligent logic in the process by the controllers in charge of handling the requests as the data is managed by the database hosted on the same virtual server or can be hosted on another server increasing the security of the IoT system [6].

Cloud native application development is based on connecting a client-server architecture for microservices by using API. The advantages that this provides are the flexibility and ease of integration, in addition to providing access to the processed and stored data needed to provide the required information [8, 13].

For the web environment, the API uses the HTTP or HTTPS protocols as appropriate, this allows requesting messages and providing a definition of the structure in the response messages. Such messages are encoded in JSON (Java Script Object Notation) object format, they are widely used because they are easy to manipulate and highly compatible with multiple languages that require the transfer of encoded data [10]. This interface must support the use of the HTTP or HTTPS protocol [7] and allow data transfer in both directions. The architecture is widely used for web platforms and in this case for IoT systems [11]. Other transfer protocols such as SFTP (Secure File Transfer Protocol) [9] can also be used to allow the transfer of other types of data such as .log files or video and audio playback formats such as .mp3 and .mp4.

### **1.1 Similar IoT Solutions**

- Utilization of an API for representational state transfer (REST), it was designed to control and coordinate the activities of e-waste collection containers, acting as a communication bridge between the collection containers and the blockchain network [7].
- IoT based geriatric care management system to achieve smart health in nursing homes, an API was used for data transmission from an external cloud platform to a LAMP server [15].
- Project to develop a responsive tool that customizes the environment through a thermal sensor and maximizes energy efficiency by combining building information modeling (BIM) data with realtime information through a weather API [16].

## **2 Description of Used Software**

The operating system supporting the IoT application is an Ubuntu server LTS version. For the integration of the IoT intelligent system, a NGINX web server was configured using a client-server architecture, using an SSL (Secure Sockets Layer) certificate to secure the HTTP protocol and transform it into HTTPS, ensuring data protection by encrypting the sending and receiving of data through the API.

The base programming language for the IoT platform is php (Hypertext Pre-Processor), we also considered the use of a design pattern to develop the software logic using MVC (Model-View-Controller) through the use of a framework called Laravel that allows the use of a refined and expressive syntax with a high compatibility in libraries for php.

A database manager called Maria DB was used to store the information collected by the IoT devices. This set of components is known as LEMP SERVER and is used for countless web platform projects, in this case it will be applied to the integration of an IoT platform two levels of headings should be numbered. Lower level headings remain unnumbered; they are formatted as run in headings.

### **2.1 IoT Smart System Architecture**

For the correct operation of the IoT system it is necessary to establish an architecture design that interlinks the elements that compose it, as shown in Fig. 1. The IoT device connects and sends telemetry through a secure port to the API, behind it are the controllers that allow the reception and response.

The variable is sent to the models through programming objects and these in turn connect to the database, allowing actions such as: create, edit and query parameters to maintain modularity in the system. Applying some intelligent logic allows hybridization between the devices that integrate it.

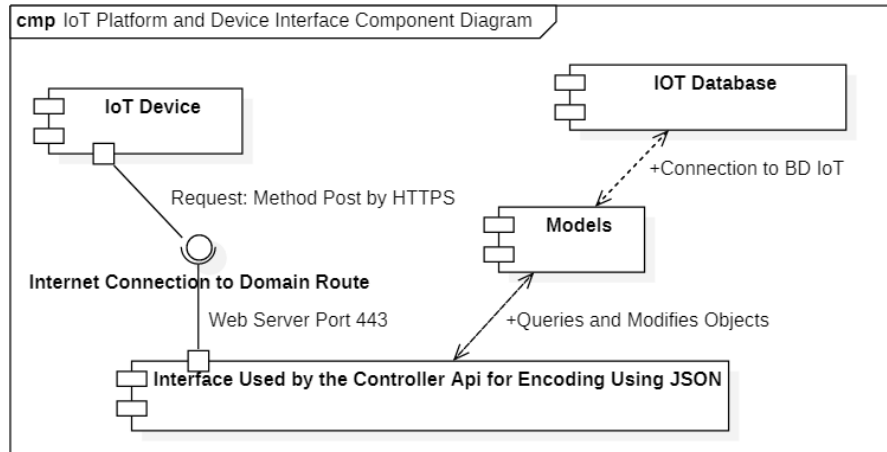


Fig. 1. Architecture between IoT Device, API Interface and IoT Server [Own creation].

## 2.2 API functionality

The operation of the interface contains a certain established logic that meets criteria such as secure communication as the data per port is encrypted, the device communicates with the interface to send and receive requests. This interface is responsible for communicating with the IoT platform through the drivers that implement methods. For this development there are the modification and status methods, the modification driver is responsible for adding information to the telemetry history also to update the last collected values and add additional instructions for the reconfiguration of the IoT device, while the status driver is the one that allows updating the IoT device when it is restarted or connected for the first time, as shown in Fig. 2.

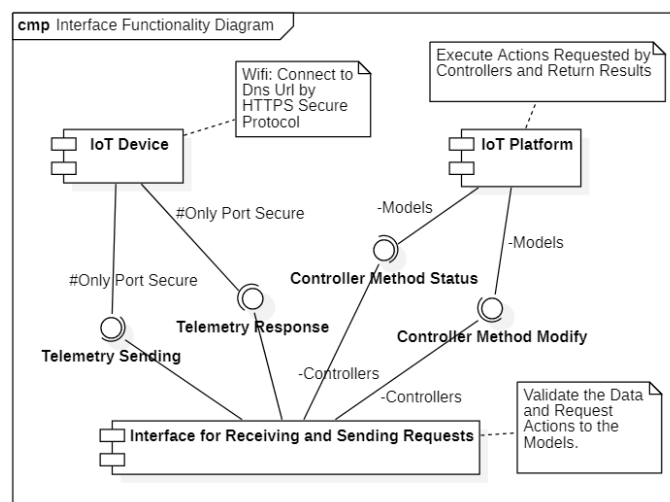


Fig. 2. IoT Device Telemetry Sending and Response Functionality [Own creation].

### 3 Analysis of Results

The verification of the functionality of the IoT intelligent system was carried out under conditions that allowed to obtain results with which we can infer the veracity of this and that such implementation is functional and multipurpose. The criteria taken into account are:

- The conditions of the cloud server are optimal.
- The IoT device was free from external agents that could cause failures.
- Internet service was stable for sending and receiving requests from the IoT device to the API.
- Establish prior registration of the IoT device, so that authentication is effective and allows to generate desired traffic.

#### 3.1 Telemetry Parameters

In sending telemetry, essential parameters must be considered in the request, this is because there is a validation to ensure that these parameters are in the database, in Fig. 3 you can see that the list of parameters includes values for authentication as well as values for sensors, type of method either status() or modify().

HOTSPOT-FJLIC / Sensor\_Modify

POST <https://hotspot.fjlic.com/api/sensor/modify>

Params Authorization Headers (9) Body Pre-request Script Tests Settings

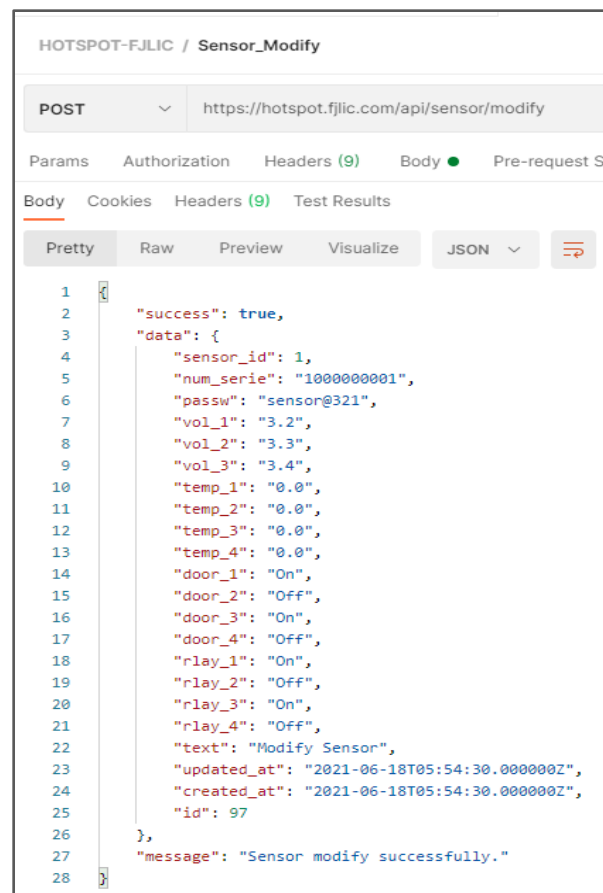
none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> num_serie	1000000001
<input checked="" type="checkbox"/> passw	sensor@321
<input checked="" type="checkbox"/> vol_1	3.2
<input checked="" type="checkbox"/> vol_2	3.3
<input checked="" type="checkbox"/> vol_3	3.4
<input checked="" type="checkbox"/> temp_1	0.0
<input checked="" type="checkbox"/> temp_2	0.0
<input checked="" type="checkbox"/> temp_3	0.0
<input checked="" type="checkbox"/> temp_4	0.0
<input checked="" type="checkbox"/> door_1	On
<input checked="" type="checkbox"/> door_2	Off
<input checked="" type="checkbox"/> door_3	On
<input checked="" type="checkbox"/> door_4	Off
<input checked="" type="checkbox"/> text	Modify Sensor

Fig. 3. Simulation of Sending Telemetry Sent by the IoT Device [Own creation].

### 3.2 API Response Consuming Methods

Behind the API, there is a method called status, this allows the IoT device to consult the last registered parameters, for this the API validates two parameters (serial number and password), which must be previously registered in the database, as shown in Fig. 4 after the query to the API, it responds with the last registered data.



The screenshot shows a REST client interface for a POST request to `https://hotspot.fjlic.com/api/sensor/modify`. The response is displayed in JSON format, showing a successful modification of a sensor.

```

1  {
2    "success": true,
3    "data": {
4      "sensor_id": 1,
5      "num_serie": "1000000001",
6      "passwd": "sensor@321",
7      "vol_1": "3.2",
8      "vol_2": "3.3",
9      "vol_3": "3.4",
10     "temp_1": "0.0",
11     "temp_2": "0.0",
12     "temp_3": "0.0",
13     "temp_4": "0.0",
14     "door_1": "On",
15     "door_2": "Off",
16     "door_3": "On",
17     "door_4": "Off",
18     "rlay_1": "On",
19     "rlay_2": "Off",
20     "rlay_3": "On",
21     "rlay_4": "Off",
22     "text": "Modify Sensor",
23     "updated_at": "2021-06-18T05:54:30.000000Z",
24     "created_at": "2021-06-18T05:54:30.000000Z",
25     "id": 97
26   },
27   "message": "Sensor modify successfully."
28 }

```

**Fig. 4.** Query by Using the Modify() Method. [Own creation]

### 3.3 IoT Device Responses

In the IoT device, the telemetry sending and API response were checked, as shown in Fig. 5, the sequence of the request is cyclical where first the IoT device consumes the API and this in turn responds with the recorded information in addition to receiving the reconfigurable parameters.

```

-----Petición ESP32 - Modify-----
{'door_2': 'Off', 'door_3': 'Off', 'door_1': 'Off', 'temp_3': '26.3125', 'temp_1': '26.3125', 'door_4': 'Off', 'num_serie': '100000001', 'temp_2': '26.3125'}
-----Respuesta de la Plataforma-----
{'data': {'temp_2': '26.3125', 'temp_3': '26.3125', 'vol_3': 'Off', 'temp_1': '26.3125', 'vol_2': 'Off', 'door_4': 'Off', 'num_serie': '100000001', 'updated_at': '2021-06-19T16:37:57.000000Z', 'id': 127, 'created_at': '2021-06-19T16:37:57.000000Z'}, 'success': True, 'message': 'Sensor modify successful'}
-----Petición ESP32 - Modify-----
{'door_2': 'Off', 'door_3': 'Off', 'door_1': 'Off', 'temp_3': '26.3125', 'temp_1': '26.3125', 'door_4': 'Off', 'num_serie': '100000001', 'temp_2': '26.3125'}
-----Respuesta de la Plataforma-----
{'data': {'temp_2': '26.3125', 'temp_3': '26.3125', 'vol_3': 'Off', 'temp_1': '26.3125', 'vol_2': 'Off', 'door_4': 'Off', 'num_serie': '100000001', 'updated_at': '2021-06-19T16:38:21.000000Z', 'id': 129, 'created_at': '2021-06-19T16:38:21.000000Z'}, 'success': True, 'message': 'Sensor modify successful'}
-----Petición ESP32 - Modify-----
{'door_2': 'Off', 'door_3': 'Off', 'door_1': 'Off', 'temp_3': '26.25', 'temp_1': '26.25', 'door_4': 'Off', 'num_serie': '100000001', 'temp_2': '26.25'}
-----Respuesta de la Plataforma-----
{'data': {'temp_2': '26.25', 'temp_3': '26.25', 'vol_3': 'Off', 'temp_1': '26.25', 'vol_2': 'Off', 'door_4': 'Off', 'num_serie': '100000001', 'updated_at': '2021-06-19T16:38:33.000000Z', 'id': 130, 'created_at': '2021-06-19T16:38:33.000000Z'}, 'success': True, 'message': 'Sensor modify successful'}

```

**Fig. 5.** IoT Device Sending and Response Frames Consuming the API Interface [Own creation].

## 4 Conclusions

The implementations in the development of IoT based systems are very popular nowadays, to carry out the development of a certain system can be a great challenge if you do not have the necessary knowledge and resources [17], the technologies that integrate an IoT system can be very varied and will depend on the adaptability of the hardware and software, to increase its potential by adding more functionality and interconnecting them with cloud services [18]. The use of API as an interface to communicate electronic devices with the cloud has great advantages such as:

- Knowing the current status of the equipment by having a telemetry history.
- Processing telemetry for analysis and behavioral analysis.
- Applying Artificial Intelligence to the data.
- Applying machine learning to the data.

With this project we sought to demonstrate that today the intercommunication of devices to the Internet is possible, as a result of the solution we obtained a minimalist API with two simple methods that are sufficient to interconnect the devices with the cloud, in addition the system was provided with security to maintain data transfer, this gave as a solution a simple architecture and that serves as a basis of understanding for the management and use of interfaces, which are widely used to intercommunicate systems with different functionalities, obtaining valuable data [19].

The realization of these solutions is becoming more and more common, as they are constantly being improved and their adaptability increases over time. The integration of technologies used in different fields or of any kind is what makes IoT unique.

## References

1. Abba Ari, Ado Adamou, Olga Kengni Ngangmo, Chafiq Titouna, Ousmane Thiare, Kolyang, Alidou Mohamadou, Abdelhak Mourad Gueroui: Enabling privacy and security in Cloud of Things: Architecture, applications, security & privacy challenges. *Applied Computing and Informatics* (2020) <https://doi.org/10.1016/j.aci.2019.11.005>.
2. Balogun, Habeeb, Hafiz Alaka, Christian Nnaemeka Egwim: Boruta-grid-search least square support vector machine for NO2 pollution prediction using big data analytics and IoT emission sensors. *Applied Computing and Informatics* (2021) <https://doi.org/10.1108/ACI-04-2021-0092>.
3. Mahmood, Ahlam Fadhil, Marwa Mohamad Rafea: Designing a collection of two IoT-Systems for real time health telemonitoring. *Journal of Engineering, Design and Technology* (2021) <https://doi.org/10.1108/JEDT-12-2020-0542>.
4. Banerjee, Anish, R. Ramesh Nayaka: A comprehensive overview on BIM-integrated cyber physical system architectures and practices in the architecture, engineering and construction industry. *Construction Innovation* (2021) <https://doi.org/10.1108/CI-02-2021-0029>.
5. Aziez, Meriem, Saber Benharzallah, Hammadi Bennoui : A full comparison study of service discovery approaches for Internet of things. *International Journal of Pervasive Computing and Communications* 15, n.º 1:30–56 (2019) <https://doi.org/10.1108/IJPCC-04-2019-0038>.
6. Tsang, Y.P., K.L. Choy, C.H. Wu, G.T.S. Ho, Cathy H.Y. Lam, P.S. Koo: An Internet of Things (IoT)-based risk monitoring system for managing cold supply chain risks. *Industrial Management & Data Systems* 118, n.º 7:1432–62 (2018) <https://doi.org/10.1108/IMDS-09-2017-0384>.
7. Sahoo, Swagatika, Arnab Mukherjee, Raju Halder: A unified blockchain-based platform for global e-waste management. In.: *International Journal of Web Information Systems* 17, n.º 5:449–79 (2021) <https://doi.org/10.1108/IJWIS-03-2021-0024>.
8. Anthony Jnr, Bokolo, Sobah Abbas Petersen, Dirk Ahlers, John Krogstie: Big data driven multi-tier architecture for electric mobility as a service in smart cities: A design science approach. *International Journal of Energy Sector Management* 14, n.º 5:1023–47 (2020) <https://doi.org/10.1108/IJESM-08-2019-0001>.
9. Halmetoja, Esa: The conditions data model supporting building information models in facility management. *Facilities* 37, n.º 7/8:484–501 (2019) <https://doi.org/10.1108/F-11-2017-0112>.
10. Varthis, Evangelos, Marios Poulos, Ilias Giarenis, Sozon Papavlasopoulos: A novel framework for delivering static search capabilities to large textual corpora directly on the Web domain: an implementation for Migne's *Patrologia Graeca*. *International Journal of Web Information Systems* 17, n.º 3:153–86 (2021) <https://doi.org/10.1108/IJWIS-10-2020-0062>.
11. Chondamrongkul, Nacha: Model-driven framework to support evolution of mobile applications in multi-cloud environments. *International Journal of Pervasive Computing and Communications* 12, n.º 3: 332–51 (2016) <https://doi.org/10.1108/IJPCC-01-2016-0003>.
12. Barroca Filho, Itamir de Moraes, Gibeon Soares Aquino Júnior: Development of mobile applications from existing Web-based enterprise systems. *International Journal of Web Information Systems* 11, n.º 2:162–82 (2015) <https://doi.org/10.1108/IJWIS-11-2014-0041>.
13. Kotciuba, Igor Yurievich, Alexey Nikolaevich Shikov, Yuri Voitekhovsky: Recommendation system for finding improved coworking area based on intelligent information technologies. *World Journal of Engineering* 18, n.º 4:621–29 (2021) <https://doi.org/10.1108/WJE-10-2020-0518>.

14. Adams Jr, Richard Manly: Overcoming disintermediation: a call for librarians to learn to use web service APIs. *Library Hi Tech* 36, n.º 1:180–90 (2018) <https://doi.org/10.1108/LHT-03-2017-0056>.
15. Tang, Valerie, K.L. Choy, G.T.S. Ho, H.Y. Lam, Y.P. Tsang: An IoMT-based geriatric care management system for achieving smart health in nursing homes. *Industrial Management & Data Systems* 119, n.º 8: 1819–40 (2019) <https://doi.org/10.1108/IMDS-01-2019-0024>.
16. Birgonul, Zeynep: A receptive-responsive tool for customizing occupant's thermal comfort and maximizing energy efficiency by blending BIM data with real-time information. *Smart and Sustainable Built Environment* (2021) <https://doi.org/10.1108/SASBE-11-2020-0175>.
17. Golondrino, G. E. C., Alarcón, M. A. O., Muñoz, W. Y. C.: Sistema IoT para el seguimiento y análisis de la intensidad de luz en plantas de interiores. 11 (2020)
18. Gómez, A. P., Cahuich, A. C. S., Gómez, J. J.: Plataforma de gestión IoT mediante técnicas de industria 4.0 para agricultura de precisión. 13 (2020)
19. Cariño, L. B. C., Morán, C. O. G.: Diseño y construcción de sensor de humedad IoT hecho de PVDF para aplicaciones en la industria del cultivo. 11 (2020)