Received June 28, 2018, accepted August 19, 2018, date of publication August 29, 2018, date of current version September 21, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2867794

Controlled and Secure Access to Promote the Industrial Internet of Things

MARCO A. MÁRQUEZ[1], REYES S. HERRERA[1], ANDRÉS MEJÍAS[1], FRANCISCO ESQUEMBRE[2], AND JOSÉ M. ANDÚJAR[1], (Senior Member, IEEE)
[1]Engineering Technical School, University of Huelva, 21819 Huelva, Spain
[2]Mathematics Department, University of Murcia, 30003 Murcia, Spain

Corresponding author: Reyes S. Herrera (reyes.sanchez@die.uhu.es)

ABSTRACT Internet of Things (IoT) aims at facilitating access to all devices that are connected to the internet, both wired and wireless. This scenario can initially seem interesting. Nevertheless, it has a lack of privacy and danger of malicious interaction with the devices. Therefore, IoT, as it stands, is not suitable for companies to make their data and devices accessible through the internet, since they could find an untidy cloud, made up of devices without the necessary control of use. This paper proposes the use of the cloud computing advantages to develop a secure access global system based on a cloud. The company will decide the controlled access to the chosen devices and data, both by employees and external people. The developed system can be used from different scenarios such as: a public cloud; an Infrastructure as a Service (IaaS); and a private cloud. To illustrate the operation of the developed system, a representative network of heterogeneous multiprotocol devices has been designed.

INDEX TERMS Internet of Things, industrial Internet of Things, secure access, fog computing, middleware architecture, service oriented interface.

LIST OF ACRONYMS

API	Application Programming Interface
CC	Cloud Computing
CoAP	Constrained Application Protocol
CS	Cloud Service
DDoS	Distributed Denial-of-Service
EJS	Easy Java Simulations
GIoT	Gate Internet of Things
HB access	Http Based access
IaaS	Infrastructure as a Service
IoT	Internet of Things
IIoT	Industrial Internet of Things
IIoTGS	Industrial Internet of Things Global System
IMAP	Internet Message Access
IP	Internet Protocol
JSON	JavaScript Object Notation
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LTI	Learning Tools Interoperability
M2M	Machine to Machine
M2P	Machine to People
MQTT	Message Queuing Telemetry Transport
OS	Operating System
OSI	Open Systems Interconnection
PLC	Programmable Logical Controllers
POP3	Post Office Protocol
PS access	Direct access from the proprietoral application to the devices in the fog
RFID	Radio Frequency Identification
REST	REpresentational State Transfer
SCADA	Supervisory Control And Data Acquisition
TCP/IP	Transmission Control Protocol / Internet Protocol
ULC	User Links Constructor
URL	Uniform Resource Locator

I. INTRODUCTION

Over the last few years, the traditional network generic infrastructure has expanded to appliances, displays, vehicles, healthcare devices, energy control devices, automotive appliances, traffic management, etc. This expansion has been possible thanks to the great technological advance experienced by sensors, actuators, microcontrollers and devices with embedded computing capability.

Nowadays people live permanently connected to the internet. This fact makes that almost any device is connected, Figure 1. As a result, the internet transcends the professional scope and is present in all areas of life. This has brought about the continuous increase of devices with connection capacity, increasingly smaller sizes and with lower consumption. For example low-power wireless devices [1] or passive radio frequency identification (RFID) tags, [2], [3].

FIGURE 1. **The cloud in the Internet of Things.**

The evolution indicated above has driven the appearance of IoT, [4].

Moreover, nowadays people manage a large part of their professional, personal and leisure information through the internet, within a cloud (Cloud Computing, CC), [5]. In this sense, IoT aims to facilitate the access to all the devices that are connected to the internet, both via cable or wireless. IoT itself is a developing technology which has its own protocols and criteria for design and development.

Manufacturers are one of the main performers in this framework. Thus, they are incorporating network interfaces in their devices, which make them convergent devices (this is the way how the devices accessible through the internet will be referred in the paper). Thus, within IoT, thousands of sensors, actuators and, in general, any connected device, can be publicly accessed by everyone. Manufacturers also provide the software needed to make their devices easily accessible through the internet (proprietary software).

This scenario can initially look interesting, even ideal. Nevertheless, it lacks privacy and is at risk of malicious interaction with the devices, [6]. Moreover, in the professional field, the devices in IoT could be exposed to malicious tracking which could result in intrusions in the company's activity. Even the accessible devices could be blocked with malicious distributed denial-of-service (DDoS) attacks. In addition, they could be saturated due to multiple concurrent attempts of access, even though they are not malicious but only intended to obtain information. In fact, DDoS attacks could also change the behavior of physical systems, such as indiscriminately controlling the sequences at traffic lights, opening/closing doors, sounding alarms, etc.

Therefore, as it stands, IoT is not suitable for companies to make their data and devices accessible through the internet, since they could find an untidy cloud, made up of devices without the necessary control of use. However, a controlled cloud could be developed, so that the company could choose among all possible data and devices, to be accessible or not (within that cloud). This new structure would allow secure access to information. The data access would be controlled according to the profiles assigned. This cloud would facilitate the expansion of the IoT into the professional area (industrial and services), where access control to the data and devices is vital. Moreover, a suitable access to the data through the internet makes the companies more efficient and allows them to generate more business value. This expansion of the IoT into the professional area is a global tendency which has been evolving over the last few years, and is referred to as the industrial internet of things (IIoT). Currently, the most common applications in this context are related to predictive maintenance. However, it is expected that the IIoT will experience a huge growth over the next years, [7], [8].

This paper proposes the use of the CC advantages (cost saving, cost based on use, resources scalability, simplification of tasks related to infrastructure management, storage and massive data processing capabilities) to develop a secure access global system based on a cloud, [9], [10]. The company will decide the controlled access to the chosen devices and data, both by employees and external personnel. It would therefore allow the proposed system to be used from different scenarios: a public cloud, an IaaS and a private cloud.

In what follows, the developed system will be referred to as IIoT global system (IIoTGS). Among other features, it allows the definition of user profiles to control the selective access to the different devices and data. IIoTGS also allows the connection of different local area networks (LANs) with distributed devices and data storage systems. In addition, it solves the problems arising from the coexistence of proprietary software with others developed in open platforms. Finally, the communication process between the user and the convergent devices becomes more transparent. The use of IIoTGS would boost competition and would increase the adoption rate of IoT in the industry.

This paper is organized as follows: in section II, the current IoT communications structures are analyzed. In section III, a general convergent multiprotocol scenario is presented. Based on the two previous sections, in section IV, the proposed system is presented, fitting it into the structures explained in section II and dividing the access into two types (http access and access with proprietary software). Section IV includes 4 subsections: the first deals with the way users access the cloud, the second presents the proposed tunnel service for accessing with proprietary software, the third describes the communications technology of the proposed system and, in the fourth, security and privacy issues are briefly discussed. Section V is devoted to illustrate, by a practical case (a representative network of heterogeneous multiprotocol devices), the operation of the developed IIoTGS. Finally, section VI provides some conclusions.

II. CURRENT IOT COMMUNICATIONS STRUCTURES

The development of IoT systems to interconnect smart objects is a complex task. Firstly, the smart objects are designed by different manufacturers. Secondly, each manufacturer provides its own proprietary software. Thus, their interoperability is a challenge. In addition, sensors, actuators and controllers have constraints in bandwidth, power, size and location. These constraints considerably affect security and privacy issues.

Some of the most common architectures could help the design and creation of IoT systems, as the open systems interconnection (OSI) model or the transmission control protocol / internet protocol (TCP/IP) model. However, a simpler approach is emerging to connect smart objects. Each level of expanded connectivity has a different set of design issues and requirements for security and privacy to be considered.

For example, in the lowest level, IoT solutions often support one smart object connecting directly to another via wireless protocol such as Bluetooth or Zigbee (device-to-device, Figure 2), [11]. As seen in figure 2, the connection is carried out in the network layer.

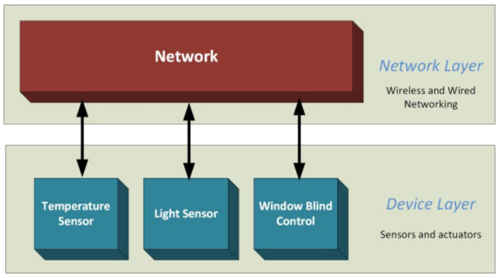


FIGURE 2. Device-to-Device architecture.

This structure provides the connectivity between small smart objects and it is suitable in personal or local settings. For example, it is good for connecting a smartwatch to the smartphone. However, when rising in the communications level, the IoT device connects directly to an internet cloud service through a local network, using traditional wired Ethernet or Wi-Fi connections. This way of connecting is named device-to-network-to-cloud communication model (Figure 3). As seen in Figure 3, the final connection of all the devices is carried out in the cloud layer through several IP (Internet Protocol) networks. It allows the exchange of data and control messages. This model amplifies the scope of the device-to-device model and makes possible the inclusion of the device in the internet. Therefore, with the structure device-to-network-to-cloud, convergent devices can be connected to the cloud layer.

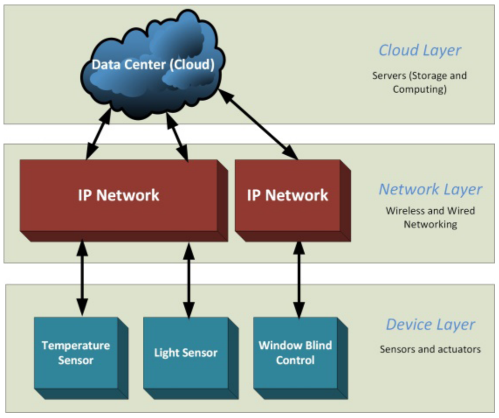


FIGURE 3. Device-to-Network-to-Cloud architecture.

The disadvantage of the device-to-network-to-cloud architecture is that all the information has to travel through the network layer because all the information is processed in the cloud layer. However, many smart devices such as fitness trackers, are not IP-enabled and do not have the native ability to connect directly to the cloud. For these devices, the application software must be operating on a local gateway device which acts as an intermediary between the device and the cloud layer (device-to-gateway-to-cloud, Figure 4). As seen in figure 4, gateway devices form the Edge/Fog layer. In addition, on the one hand, the gateway may also provide security. For example, the smartphone would act as gateway between the smartwatch and the cloud layer. On the other hand, the fog layer allows to decrease the traffic through the network layer, [12].

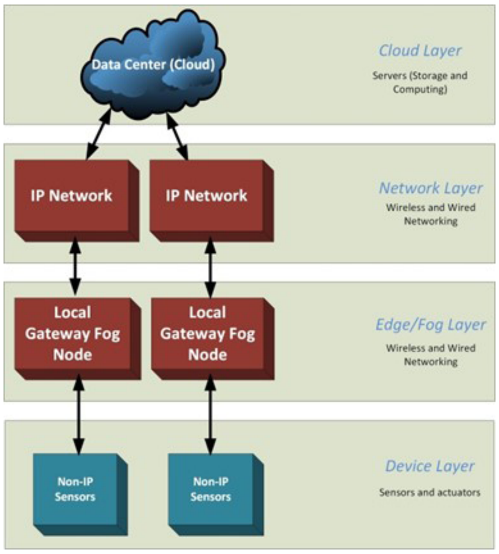


FIGURE 4. Device-to-Gateway-to-Cloud architecture.

To complete the model, in the highest communications level, another connection option supports smart device data collection and transfers through a gateway to a local IP network (device-to-gateway-to-cloud-to-application, Figure 5). The data flow to the cloud and is then available for users to

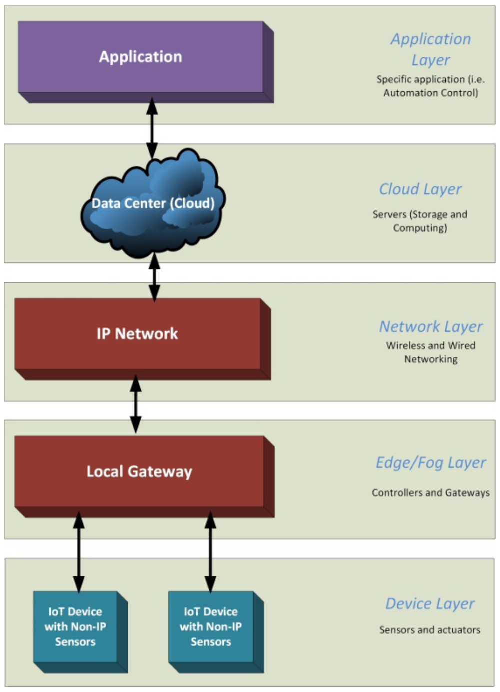

Application
Application Layer
Specific application (i.e. Automation Control)
Data Center (Cloud)
Cloud Layer
Servers (Storage and Computing)
IP Network
Network Layer
Wireless and Wired Networking
Local Gateway
Edge/Fog Layer
Controllers and Gateways
IoT Device with Non-IP Sensors
IoT Device with Non-IP Sensors
Device Layer
Sensors and actuators

FIGURE 5. Device-to-Gateway-to-Cloud-to-Application architecture.

export and analyze in the application layer. the data are often analyzed in combination with data from other sources or other clouds.

III. GENERAL CONVERGENT MULTIPROTOCOL SCENARIO

As discussed in the introduction, both in IoT and IIoT frameworks, manufacturers of convergent devices provide proprietary software for their products' connection. This software includes the application needed to access it through the internet.

In contrast, open platforms (Raspberry Pi, [13], Arduino, [14], etc.) have greatly increased for their use, for example, in smart cities, [15]. It has made them suitable to implement low cost devices accessible through the internet, with analogous characteristics to the most expensive proprietary devices. Note that the development of open platforms allows the use of free software.

One of the main characteristics of the open platforms is the great variety of protocols that can be used. They range from those used in the industrial field, such as Modbus, [16], to those used in IoT, such as message queuing telemetry transport (MQTT) or REST (representational state transfer), or those used in low-level communications, such as WebSocket and Socket, etc.

Another additional advantage is the huge amount of information about the open platforms which can be found on the internet. It facilitates the development of new software and the growth of this kind of technology. Because of the fast expansion and acceptance of the open platforms, manufacturers are beginning to include them in their catalogue of products. For example, there are Arduino IDE programmable PLCs in Siemens', [17], or in Industrial Shields' [18] catalogues.

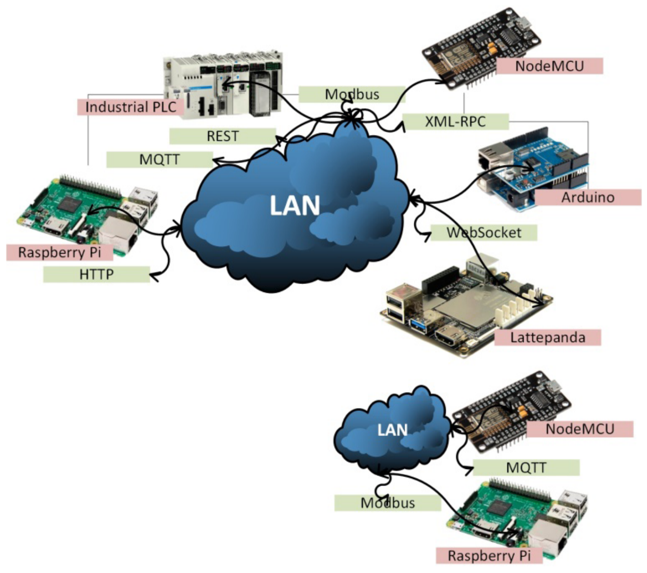


FIGURE 6. Network of heterogeneous multiprotocol devices.

IV. PROPOSED GLOBAL SYSTEM ARCHITECTURE

The IIoTGS proposed in this paper is applicable to the most general and, probably, complicated scenario, i.e., multiprotocol. In the case of commercial devices, the communication architecture is imposed by the proprietary software. Therefore, the function of the IIoTGS is to connect the user interface running on his/her computer to the convergent device connected to a LAN. Nevertheless, in the case of those devices whose connection is developed within open platforms, the design of an own communication architecture is included in the functions of the IIoTGS.

Figure 7 shows the developed IIoTGS architecture. It is based on one of the current IoT structures explained in Section II, specifically device-to-gateway-to-cloud, Figure 4. Thus, it includes four layers: device, fog, network and cloud. In the proposed architecture, the gateway of the device-to-gateway-to-cloud architecture incorporates new functions, as explained in the present section. The gateway in the proposed architecture is represented in figure 7 as GIoT (Gate IoT). In addition, a new element called cloud services (CS) has been included; its function will also be explained in this section.

As seen in Figure 7, the device layer represents a multiprotocol scenario. Each convergent device is directly related to a controller to allow the access to its data through the LAN.

The controllers in Figure 7 form the fog layer and carry out fog computing (fog networking or simply fogging), [19]. They collect data from the sensors and send signals to other controllers according to such collected data. This is the machine to machine (M2M) communications mode. In this scenario, sensors, actuators and controllers exist within the fog; i.e., information is not necessarily sent beyond the network layer. The fog layer sorts the data sent to the cloud and

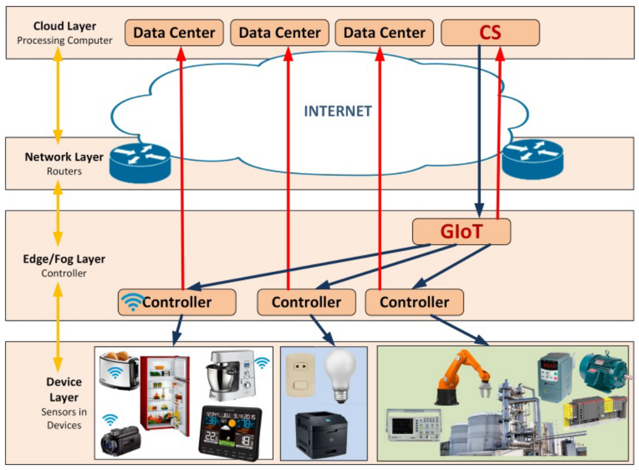


FIGURE 7. **IIoTGS Communications architecture.**

increases the system performance, while the traffic through the network layer decreases, [20]–[23]. The controllers in the fog layer can be implemented by open platforms like Raspberry Pi or Arduino, among others. Some of them are shown in the multiprotocol scenario of Figure 6.

In other cases, in which the analysis of data from several routers is necessary, or simply the communication must be machine to people (M2P), the IP-enabled controller forwards information across the network layer and allows users to access the controller remotely. In addition to forwarding basic information in an M2M configuration, some controllers are able to perform more complex operations.

In fact, any event in a device can produce a M2M or M2P communication in the fog layer and the corresponding service is sent to the cloud layer only if necessary.

The network layer devices are routers and data centers. To optimize the operation of data centers, one or more local routers are used. They are the interface between the LAN and the internet.

In the described scenario, the different elements are presented in an unsorted cloud. They are not interrelated. There are no user profiles to filter the criteria to access the information. The cloud users can access all those configurable resources, whose data can be stored and reviewed with minimal computational costs.

The high availability and expandability of the data in the cloud makes this service an excellent way to increase the efficiency of the IoT systems.

Data storing and analyzing can be done in the cloud instead of in the IoT devices. Thereby all the services are in the cloud. Hence, data and resources are always available for all the devices connected to the internet.

In the IIoTGS, besides the advantages of the IoT presented above, the introduction of the GIoT in the fog layer is proposed. It publishes the access to the controllers in the cloud in a sorted, controlled and transparent way and independently of the used communication protocol (proprietorial software or open platforms).

IIoTGS also introduces a new service in the cloud, CS, which incorporates the necessary protocol to establish the communication between the GIoT and the cloud. Thus, the GIoT makes public in the cloud layer (using the CS) the fog layer devices, through the internet. This will be explained in detail in subsection IV-C.

To do that, two possible communication scenarios are considered: the first presents capability of protocol implementation (corresponding to the systems designed within open platforms); and the second is restricted to the protocol chosen by the manufacturer (these are the convergent devices with proprietorial software).

In the first case, the chosen protocol is the web technology (XMl-RPC, SSE, WebSocket, REST, etc.); the links between the user computer and the fog layer are identified by the corresponding URL (Uniform Resource Locator).

As is well known, HTTP-based protocols allow traffic to be easily forwarded by the URL. This case will be called Http Based (HB). This access also includes those protocols easily translatable to http, as constrained application protocol (CoAP), and those transportable in WebSocket, as MQTT.

In the second case, the access is to a controller with proprietary software. The variety of protocols used by these controllers is very wide. Thus, the communication channels are implemented by tunnels as will be explained in detail in subsection IV-C. In what follows, this case will be called Proprietary Software (PS).

Regarding the communication architecture, IIoTGS uses a middleware layer to make transparent the communication process between the user computer and the convergent device [24]–[28].

Middleware works as a software distributed abstraction layer. It is located between the application and the lower layers (operating system, OS, and network layer).

Therefore, the software developed in the middleware layer provides an application programming interface (API). The API hides the complexity of the general communication problem. It also hides the low-level programming corresponding to the direct access by means of the proprietary software.

IIoTGS presents two principal differences regarding current IoT architecture. On the one hand, it creates a service in the cloud, CS, to allow the direct access from the proprietorial application to the devices in the fog (PS access). Remember that HB access is also supported by IoT architecture. On the other hand, IIoTGS includes an element (GIoT) in the fog layer to register in the cloud IaaS both kinds of access: HB and PS. GIoT makes the control of the access from the outside according to the corresponding user profiles. However, GIoT does not limit the direct access to the internet from the controllers in the fog layer. So, events produced in the production line of a company can make use of the services in the cloud without being controlled by GIoT (please see Figure 7).

Below, two technological issues which are important for the design of the IIoTGS are addressed: constrains imposed by the browsers and tunnels technology.

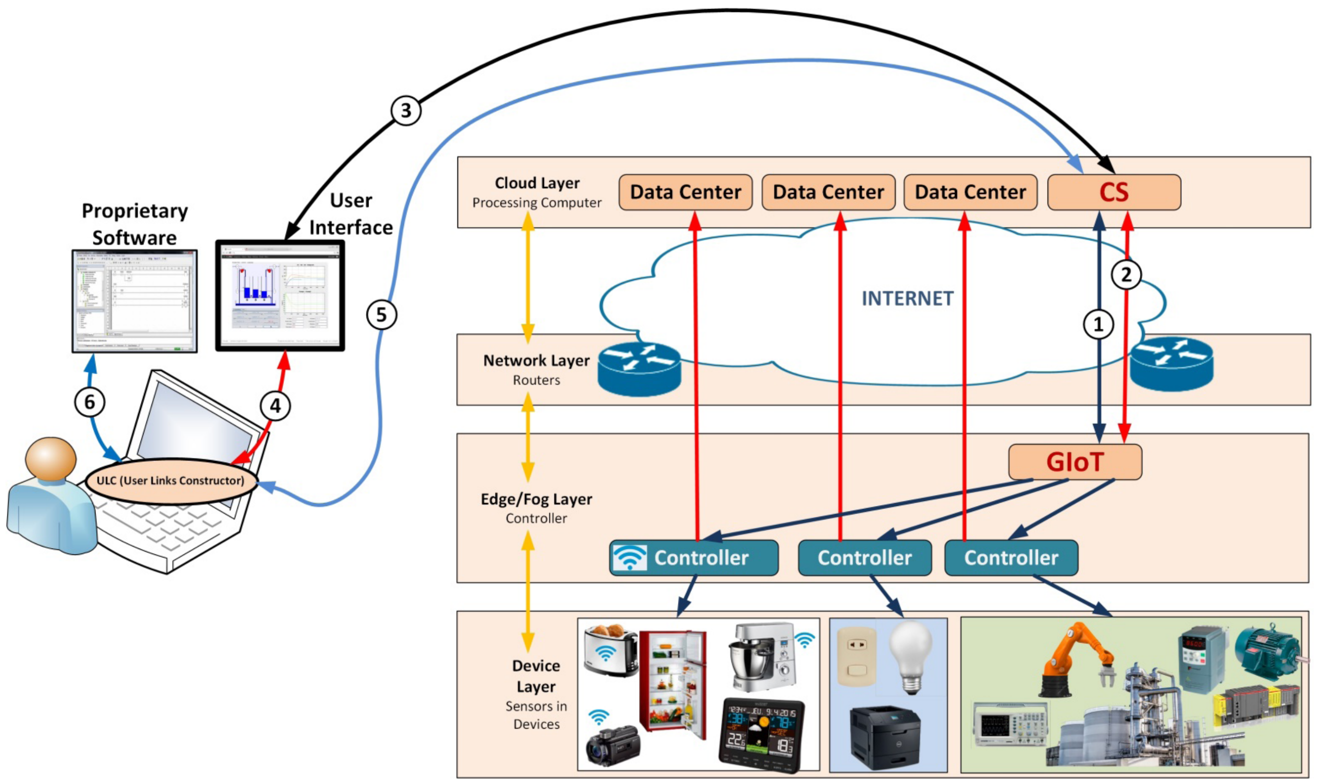


FIGURE 8. IIoTGS Communications Architecture.

A. ACCESS TO THE CLOUD BY THE USER

Access to convergent devices from the internet is carried out from the user computer. In this paper, the necessary application is called user interface (UI). In the most general case, the user interface is developed in the browser framework. IIoTGS proposes the use of the UI in the browser framework although the convergent controller is commercial and has proprietary software as access application. This is not a trivial matter, as explained below.

Due to security issues, in the latest versions of the browsers their communication capabilities have been limited to HB access. For example, in order to eliminate its own capability of interaction with the physical system in which it runs, the browser does not allow the running of Java applets to reproduce the client-server architecture. This fact presents the additional consequence of losing the capability of creating connections between the computer where the browser is running and the cloud where the convergent device is accessible.

To keep the connection capability between the browser and the convergent device, the applications programmer can use a JavaScript engine that allows creating HB connections. This is the proposed procedure by IIoTGS to implement the UI if the controller is developed within open platforms. In the case of a controller with proprietorial software, the connection between its software and the controller is also necessary. In addition, the connections must be transparent for the user.

However, the lack of local running capability of the browser makes it impossible to call the operating system and any software installed on the computer. Therefore, using JavaScript for directly accessing the controllers with proprietorial software is not possible.

B. THE SERVICE OF TUNNELS

In a network of heterogeneous multiprotocol devices, one of the most efficient ways of creating the connections between the proprietary software and the convergent devices in the fog is the tunnel. In addition, this kind of connection is transparent for the programmer and independent of the communication protocol used, [29].

Note that the internet is composed of a set of interconnected LANs, managed by different entities. Each autonomous system has its own policy, called traffic security policy. The traffic that manages to overcome the security policies of an autonomous system is called friendly traffic. In this sense, the use of tunnels provides, besides the transparency indicated above, the possibility of overcoming the security policies of each autonomous internet system, [30].

The tunnels forward all the traffic encapsulated in the same way, independently of its protocol. They also facilitate compliance with all traffic safety policies throughout the different autonomous systems on the internet. An additional advantage of using tunnels (exploited by IIoTGS) is the possibility of encrypting the data to protect their content in the transit through the internet, [31].

Each information transit circuit established by a tunnel developed through the internet, is completely defined by the input point of the tunnel in the node that promotes the communication and the output point in the LAN. Thus, each connection is defined by the pair consisting of the input point in the user's computer and the output point in the fog layer.

C. IIoTGS COMMUNICATIONS TECHNOLOGY

From here on, the mechanisms used to create the necessary channels for the different connections are presented. These mechanisms are hidden for the programmer by the middleware layer.

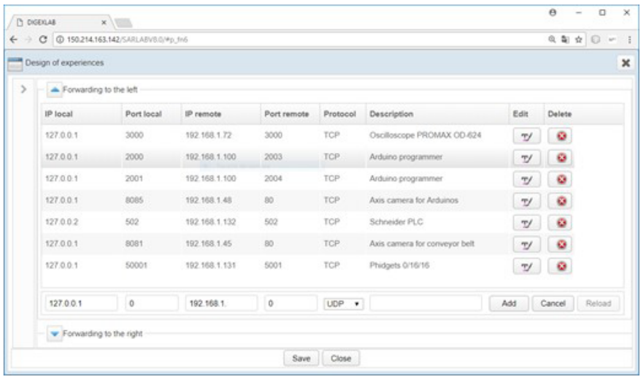
GIoT creates two tunnels between the CS and itself to send the information that reaches CS to the corresponding controller in the fog layer (see Figure 7). The first tunnel (blue arrow between CS and GIoT in Figure 7, 1 in Figure 8) allows the HB access through a URL that refers to the corresponding controller. The second one (red arrow between CS and GIoT in Figure 7, 2 in Figure 8) is the tunnel that PS access need. In both cases, the information in the tunnels is encrypted. Figure 8 presents Figure 7 completed with the user's computer to illustrate all the communication channels and is explained below.

Therefore, the connections between the user's computer and the CS are as follows:

1. The user interface (in the browser) can be connected directly to the convergent device using any HB protocol. This is the case of a non-commercial device, or the case of a commercial device capable of using the HB protocol (like an IP camera), 3 in Figure 8.
2. To access the cloud from the user's browser through a proprietary software installed on his/her computer (PS access), this paper proposes the installation of a software service (daemon) on the user's computer. This software is called user links constructor (ULC). ULC receives browser commands through a WebSocket connection, (4 in Figure 8). Therefore, the ULC has two functions: to open the necessary communication channels between the user's computer and the cloud (5 in Figure 8); and to run the necessary proprietorial software (6 in Figure 8). To do that, ULC provides the API corresponding to the middleware layer. The browser accesses this API through a WebSocket channel (4 in Figure 8), through which the user sends the commands encoded in JavaScript object notation (JSON). In summary, ULC is the link between the user's computer, its browser, the proprietary software and the cloud where the controller is accessible.

In addition, the API offers the possibility of installing applications, resident on the internet, on the user's computer as well as detecting updates. In this way, proprietary software does not need to be installed initially in the user's computer.

In summary, the IIoTGS works in the following way: GIoT contains a tunnels server to manage the PS access and a proxy server to manage HB access. In addition, GIoT publishes the controllers in the IaaS cloud (using the CS) to make them visible in the internet and accessible in an organized way. The controllers can be, for example, the equipment of a



IP local	Port local	IP remote	Port remote	Protocol	Description	Edit	Delete
127.0.0.1	3000	192.168.1.72	3000	TCP	Oscilloscope PROMAX OD-624		
127.0.0.1	2000	192.168.1.100	2003	TCP	Arduino programmer		
127.0.0.1	2001	192.168.1.100	2004	TCP	Arduino programmer		
127.0.0.1	8085	192.168.1.48	80	TCP	Axis camera for Arduinos		
127.0.0.2	502	192.168.1.132	502	TCP	Schneider PLC		
127.0.0.1	8081	192.168.1.45	80	TCP	Axis camera for conveyor belt		
127.0.0.1	50001	192.168.1.131	5001	TCP	Phidgets 0/16/16		
127.0.0.1	0	192.168.1.	0	UDP			

FIGURE 9. Interface to configure the GIoT.

factory. The user must install the ULC on his/her computer. ULC requests access to those controllers through the CS. The communications between both are controlled by GIoT. The ULC contacts the CS and the GIoT creates the corresponding connections in the CS between the user's computer and the controllers in the fog layer, Figure 8. Once the channels are established, ULC activates the proprietary software to interact with the controllers.

In addition, the IIoTGS provides a database to define different connection topologies according to the user's profile. In this way, the system administrator can configure different user profiles.

To manage the database, a web interface has been created to assign the possible connections and competences of the different user profiles to access the controllers, Figure 9. This interface can be integrated in the company's business application.

D. SECURITY AND PRIVACY IN IIoTGS

GIoT has been implemented on an application server, which provides the communication servers mentioned above (tunnel server and proxy). To control HB access, a container provided filter has been implemented above the proxy server. If access is requested through a GET request, the filter allows security information to be obtained from the URL corresponding to the request. The URL incorporates the necessary information to control the access.

Thus, the URL includes, besides the path, the user identification by means of parameters, as for example user and password, key, etc. Parameters can indicate an external server to perform identification through a protocol as the internet message access protocol (IMAP), Lightweight Directory Access Protocol (LDAP) or post office protocol (POP3), among others. The container provided filter also allows the validation of POST requests. In this case, the parameters can be obtained from the own request. An example of this method is the learning tools interoperability (LTI) protocol.

To control the PS access, the tunnel server has been used. This server can be implemented with any protocol according to the document RFC 6169, [32]. In fact, a security layer has been integrated in this server and customized using the next two interfaces: PasswordAuthenticator for password-based authentication and PublickeyAuthenticator for key based authentication. The access by delegated server has also been implemented by means of the PasswordAuthenticator.

Therefore, according to the information explained in this section, IIoTGS can achieve the following goals:

- Access through the internet to devices connected to different LAN.
- The creation of sets of convergent devices with unified access control.
- Access to several commercial devices with proprietary software.
- The abstraction of the designer to the complexity inherent to the programming of communications.
- The overcoming of the restrictions generated by the browsers new generation.

According to the explained above, the developed system establishes the possibility of sorting and filtering the information of the production line of a factory. In this way, operating/production devices, hardware data and maintenance software, among others, can be safely accessed to retrieve information or manage the factory production process. The developed system also allows setting up a network in which data from multiple industries are collected and analysed to support the decision-making process.

The IIoT model proposed in this paper improves the manufacturing business by properly connecting people with the right information, which comes from all objects related to the manufacturing process, such as sensors that display the flows of factory operations and the supply chain. As data is collected, trends and relationships, which reveal opportunities for improvement of manufacturing processes, can be identified.

V. A PRACTICAL CASE

In this section, a practical case is presented. To illustrate the IIoTGS operation, a representative network of heterogeneous multiprotocol devices has been designed. It is made up of commercial devices with their own proprietary software, and controllers developed in open platforms. IIoTGS is able to create the communication channels between the user's computer with the corresponding profile and those controllers, in a concurrent or successive way. Regarding the user effort, he/she only must be connected to the internet anywhere in the world.

Figure 10 shows a photograph of the physical devices in the system. They can be divided into two parts. In the top left, there is a pilot plant with a conveyor belt whose movement is controlled by a commercial PLC (a Schneider Electric PLC, [33]). In the top right, there are different devices such as servomotors, DC motors, displays and LED. They are connected to a set of Arduino boards that control them. Arduino boards also process the output of two Phidget boards, [34], which produce analogical and digital signals. The location

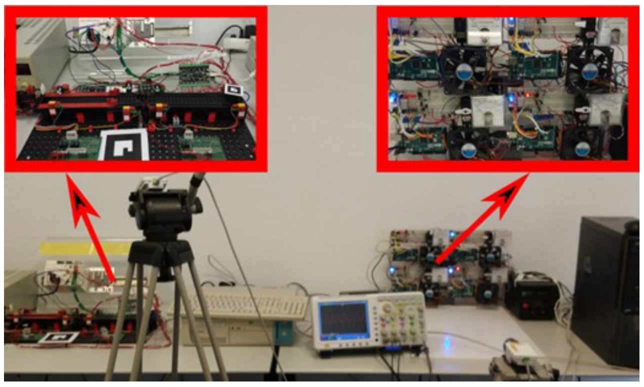


FIGURE 10. Device layer of the practical case.

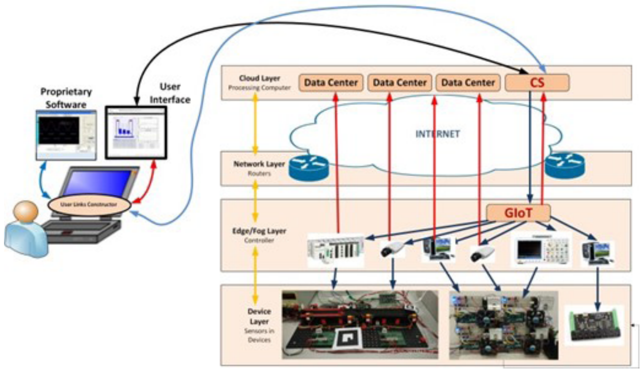


FIGURE 11. Practical case IIoTGS Architecture.

of the complete set of devices can be seen in the bottom of Figure 10.

All these devices constitute the device layer corresponding to the practical case according to Figure 8. This figure has been reedited in Figure 11, where the different layers have been adapted to the used in the practical case presented.

The device layer in Figure 11 is constituted by the devices presented in Figure 10. The fog layer is constituted as follows: the first controller is the PLC that controls the conveyor belt. To program it, the manufacturer provides a proprietorial software called Unity - Pro.

Arduino boards are connected to the second controller, a x86 processor running Windows OS. The third controller is another x86 running Linux OS, where the Phidget boards are connected. This third controller is used to generate physical signals according to the user's commands through the UI. The control program for these two controllers has been developed in the framework of the Easy Java Simulations (EJS) authoring tool application. EJS, [35], allows creating interactive simulations and graphical user interfaces. EJS has the necessary drivers to directly access the Arduino and Phidget boards, [36]. In addition, it allows the creation of the local control algorithms necessary to optimize information traffic to the cloud (which allows M2M communication between the different controllers in the fog layer). EJS also has the necessary elements to establish remote connections using WebSocket.

Another controller in the fog layer is a commercial oscilloscope (Promax OD-624 oscilloscope, [37],) which shows an analogue PWM signal from one of the Arduino boards. Promax provides proprietary software to optimize the use of the oscilloscope from LAN.

Finally, two Axis IP cameras, [38], (one facing the conveyor belt with the PLC and the other facing the set of Arduino boards) are also controllers in the fog. They are used to make their video streams accessible through the internet within IIoTGS.

Regarding the cloud layer, in the use case presented in this section, a private cloud has been used.

The GIoT element is a RaspBerry Pi (although it could be any other more complex and powerful architecture). It publishes the controllers in the cloud, installing the CS. Remember that CS provides the necessary public IP to make accessible the nodes in the LAN through the internet. In addition, GIoT defines and creates the links between itself (in the fog layer) and the CS (in the cloud layer).

The user accesses the UI through the URL from a browser, with the profile provided by the administrator. Once validated, the user can connect to the HB access capable controllers (Axis cameras and Linux or Windows x86). Accessing the cameras the user can see the video images. Accessing Windows x86 the user can program Arduino boards and/or monitor them. Accessing the Linux x86 the user can programme or monitor the Phidget boards. Arduino boards control the devices shown in the top right of Figure 10. In addition, they monitor the signals provided by the Phidget' boards. The connections described in this paragraph are HB.

To access the PLC or the oscilloscope (with proprietary software), the user must first install the ULC on his/her computer. It is available in the cloud. This software is necessary to create the needed tunnels between the user's PC and the fog layer. These are PS access. The user can programme or monitor the PLC by means of the Unity-Pro software (provided by the manufacturer). The PLC controls the movement of the conveyor belt. In addition, accessing the oscilloscope, the user can monitor the signals provided by the Arduino boards using the software provided by Promax. The connections described in this paragraph are PS.

The Unity-Pro proprietary software is privately provided by the manufacturer. However, IIoTGS automatically downloads it from the internet and installs it on the user's computer as described in section IV-C.

The administrator defines the access allowed for each user profile through the GIoT configuration software, Figure 9.

Therefore, IIoTGS allows remote access to a network of heterogeneous multiprotocol devices in a secure and controlled way. In addition, the devices can be commercially obtained (with proprietorial software) or developed within open platforms.

VI. CONCLUSION

This paper proposes the use of the cloud computing advantages to develop a secure access global system, based on a cloud, in order to promote the IIoT. The developed system facilitates the communication between "things". Things can be commercial devices (with proprietary software) or those developed within open platforms. The developed system can be considered global because it solves, in a secure way, the problem of network (internet and LAN) communications between heterogeneous multiprotocol devices. The developed system provides the technological means to access those networks. In addition, access is carried out according to the corresponding user profile. This feature overcomes the security and privacy problems of IoT and promotes its introduction in the industry: IIoT.

The IIoTGS is used in several Spanish Universities and the work to be done in the immediate future is, besides improve and optimize its operation, the development of a discovery service. I.e., the IIoTGS minimize a lot of configuration task to access the fog layer, among others, those related to the network electronics configuration. With the future discovery service, IIoTGS would automatically recognize the devices connected to the fog layer in the LAN, both from the LAN and through the internet.

REFERENCES

[1] L. Jiang, L. D. Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu, "An IoT-oriented data storage framework in cloud computing platform," *IEEE Trans Ind. Informat.*, vol. 10, no. 2, pp. 1443–1451, May 2014.

[2] G. Roussos, S. S. Duri, and C. W. Thompson, "RFID meets the Internet," *IEEE Internet Comput.*, vol. 13, no. 1, pp. 11–13, Jan. 2009.

[3] C. Floerkemeier, C. Roduner, and M. Lampe, "RFID application development with the Accada middleware platform," *IEEE Syst. J.*, vol. 1, no. 2, pp. 82–94, Dec. 2007.

[4] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[5] F. Tao, Y. Cheng, L. D. Xu, L. Zhang, and B. H. Li, "CCIoT-CMfg: Cloud computing and Internet of Things-based cloud manufacturing service system," *IEEE Trans Ind. Informat.*, vol. 10, no. 2, pp. 1435–1442, May 2014.

[6] S. Pinto, T. Gomes, J. Pereira, J. Cabral, and A. Tavares, "IIoTEED: An enhanced, trusted execution environment for industrial IoT edge devices," *IEEE Internet Comput.*, vol. 21, no. 1, pp. 40–47, Jan./Feb. 2017.

[7] J. Byun and S. Park, "Development of a self-adapting intelligent system for building energy saving and context-aware smart services," *IEEE Trans. Consum. Electron.*, vol. 57, no. 1, pp. 90–98, Feb. 2011.

[8] J.-Q. Li, F. R. Yu, G. Deng, C. Luo, Z. Ming, and Q. Yan, "Industrial Internet: A survey on the enabling technologies, applications, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1504–1526, 3rd Quart., 2017.

[9] P. P. Ray, "A survey of IoT cloud platforms," *Future Comput. Inform. J.*, vol. 1, nos. 1–2, pp. 35–46, 2016.

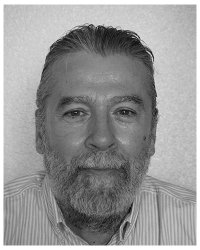
[10] G. Suciu, L. Bezdedeanu, A. Vasilescu, and V. Suciu, "Unified intelligent water management using cyberinfrastructures based on cloud computing and IoT," in *Proc. 21st Int. Conf. Control Syst. Comput. Sci. (CSCS)*, May 2017, pp. 606–611.

[11] P. Gandotra *et al.*, "A survey on device-to-device (D2D) communication: Architecture and security issues," *J. Netw. Comput. Appl.*, vol. 78, pp. 9–29, Jan. 2017.

[12] blogs@Cisco–Cisco Blogs. *Fog Computing: Bringing Cloud Capabilities Down to Earth*. Accessed: Jan. 29, 2018. [Online]. Available: https://blogs.cisco.com/digital/fog-computing-bringing-cloud-capabilities-down-to-earth

[13] *Raspberry Pi—Teach, Learn, and Make with Raspberry Pi*. Accessed: Jun. 13, 2017. [Online]. Available: https://www.raspberrypi.org/

[14] *Arduino—Home*. Accessed: Jun. 13, 2017. [Online]. Available: https://www.arduino.cc/
[15] B. Ahlgren, M. Hidell, and E. C.-H. Ngai, ''Internet of Things for smart cities: Interoperability and open data,'' *IEEE Internet Comput.*, vol. 20, no. 6, pp. 52–56, Nov./Dec. 2016.
[16] *The Modbus Organization*. Accessed: Apr. 3, 2018. [Online]. Available: http://www.modbus.org/
[17] *SIMATIC IOT2020—Siemens Automation Cooperates With Education—Siemens*. Accessed: Oct. 31, 2017. [Online]. Available: http://w3.siemens.com/mcms/sce/en/simatic-iot2020/pages/default.aspx
[18] F. Aubets, *Home-IIoT*. Accessed: Oct. 31, 2017. [Online]. Available: https://www.industrialshields.com/
[19] A. V. Dastjerdi and R. Buyya, ''Fog computing: Helping the Internet of Things realize its potential,'' *Computer*, vol. 49, no. 8, pp. 112–116, Aug. 2016.
[20] C. K. Dehury and P. K. Sahoo, ''Design and implementation of a novel service management framework for IoT devices in cloud,'' *J. Syst. Softw.*, vol. 119, pp. 149–161, Sep. 2016.
[21] S. Abdelwahab, B. Hamdaoui, M. Guizani, and T. Znati, ''Cloud of things for sensing-as-a-service: Architecture, algorithms, and use case,'' *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1099–1112, Dec. 2016.
[22] C. Wang, Z. Bi, and L. D. Xu, ''IoT and cloud computing in automation of assembly modeling systems,'' *IEEE Trans. Ind. Inform.*, vol. 10, no. 2, pp. 1426–1434, May 2014.
[23] B. W. Khoueiry and M. R. Soleymani, ''A novel machine-to-machine communication strategy using rateless coding for the Internet of Things,'' *IEEE Internet Things J.*, vol. 3, no. 6, pp. 937–950, Dec. 2016.
[24] D. Akdur and V. Garousi, ''Model-driven engineering in support of development, test and maintenance of communication middleware: An industrial case-study,'' in *Proc. Int. Conf. Model-Driven Eng. Softw. Develop.*, 2015, pp. 1–8.
[25] D. F. H. Sadok, L. L. Gomes, M. Eisenhauer, and J. Kelner, ''A middleware for industry,'' *Comput. Ind.*, vol. 71, pp. 58–76, Aug. 2015.
[26] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, ''Middleware for Internet of Things: A survey,'' *IEEE Internet Things J.*, vol. 3, no. 1, pp. 70–95, Feb. 2016.
[27] N. Cai, M. Gholami, L. Yang, and R. W. Brennan, ''Application-oriented intelligent middleware for distributed sensing and control,'' *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 947–956, Nov. 2012.
[28] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, ''IoT Middleware: A survey on issues and enabling technologies,'' *IEEE Internet Things J.*, vol. 4, no. 1, pp. 1–20, Feb. 2017.
[29] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 5th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2010.
[30] J. Hawkinson and T. Bates, ''Guidelines for creation, selection, and registration of an autonomous system (AS),'' Tech. Rep. BCP 6, RFC 1930, Mar. 1996. [Online]. Available: https://www.rfc-editor.org/info/rfc1930, doi: 10.17487/RFC1930.
[31] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli, ''Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting,'' *Comput. Netw.*, vol. 53, no. 1, pp. 81–97, 2009.
[32] S. Krishnan, D. Thaler, and J. Hoagland, ''Security concerns with IP tunneling,'' Internet Eng. Task Force (IETF), Fremont, CA, USA, Tech. Rep. RFC 6169, Apr. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6169, doi: 10.17487/RFC6169.
[33] *Global Specialist in Energy Management and Automation|Schneider Electric*. Accessed: Oct. 19, 2017. [Online]. Available: https://www.schneider-electric.co.uk/en/
[34] Phidgets Inc. *Products for USB Sensing and Control*. Accessed: Dec, 19, 2017. [Online]. Available: https://www.phidgets.com/
[35] *Welcome to Easy Java Simulations Home Page*. Accessed: Jun. 13, 2017. [Online]. Available: http://www.um.es/fem/Ejs
[36] A. Mejías, R. S. Herrera, M. A. Márquez, A. J. Calderón, I. González, and J. M. Andújar, ''Easy handling of sensors and actuators over TCP/IP networks by open source hardware/software,'' *Sensors*, vol. 17, no. 1, p. 94, 2017.
[37] *PROMAX Electronica S. L.* Accessed: Oct. 19, 2017. [Online]. Available: http://www.promaxelectronics.com/ing/
[38] *Axis Communications—Líder en cámaras de red y otras soluciones con tecnología IP | Axis Communications*. Accessed: Oct. 31, 2017. [Online]. Available: https://www.axis.com/es/es



MARCO A. MÁRQUEZ received the industrial engineering degree from the University of Seville, Seville, Spain, in 1979, and the master's degree in industrial engineering from the University of Huelva, Huelva, Spain, in 2009. He was an Associate Professor with the Department of Electronic Engineering, Computer Systems and Automatic Control, University of Huelva, from 1994 to 2002. He is currently with Huelva University. He is currently a regional Instructor at Cisco Systems in the CIT Business University Foundation, University of Cádiz, Cádiz, Spain. His research interests include new e-learning technologies and the communications aspects in remote labs. His developments are the basis UNILABS communications system, a network of Spanish universities that share its laboratories through the Internet.

REYES S. HERRERA was born in Huelva, Spain. She received the industrial engineering degree in 1995 and the Ph.D. degree in electrical engineering in 2007. She is currently an Associate Professor with the Department of Electrical Engineering, University of Huelva. Her research includes electrical power theory, electrical power quality, power conversion systems, renewable energy systems, and engineering education.

ANDRÉS MEJÍAS received the industrial engineering degree, the Dipl. degree in computing, and the master's degree in production systems engineering from the University of Seville, Seville, Spain, in 1989, 1993, and 2005, respectively, and the Ph.D. degree in industrial engineering from the University of Huelva, Huelva, Spain, in 2012. He received the award for the Best Doctoral Thesis of the IEEE Education Society in 2013 (spanish chapter).

He has been the Director of the University's Scientific Computer Centre from 1995 to 1998 and the Sub-Director of the Computing Central Service from 1998 to 1999. He is currently a Full Professor with the Department of Electronic Engineering, Computer Systems and Automatic Control, University of Huelva, Huelva, Spain. His research interests include new e-learning technologies, mainly those related to remote labs.

FRANCISCO ESQUEMBRE was born in 1963. He received the Ph.D. degree in mathematics in 1991 from the University of Murcia. Since 1986, he has been with the University of Murcia, where he is currently a Full Professor of mathematical analysis.

He began his academic career studying dynamical systems, both discrete and continuous, from a theoretical point of view. Subsequently, he devoted himself to the modeling and simulation of physical phenomena and engineering, a field in which he has developed a modeling tool called Easy Java/Javascript Simulations. This tool and its developments for teaching and research have reached a worldwide relevance being used by students, doctoral students, and professors and researchers from all over the world.

Until 2017, he has published 34 articles in indexed journals and 14 in non-indexed journals. He has authored or co-authored two books and six book chapters. He has registered three computer programs in the Intellectual Property Registry. He has given presentations at numerous international conferences, with over 15 plenary lectures and many other guests. He has participated in 7 regional projects (2 of them as IP, one on supercomputing for mathematical problems), 8 national, and 10 international.

JOSÉ M. ANDÚJAR was born in Huelva, Spain. He received the Ph.D. degree in 2000. He is currently a Full Professor of systems engineering and automatic control with the University of Huelva. He holds 12 international patents. He has over 100 papers published in indexed journals in the ISI Journal Citation Reports. Specifically, he has 51 quartile Q1 publications in 20 different journals (most of these are among the top 10 in their categories, and several are number one). He has led or co-led over 50 research projects funded by public institutions and companies. His main research interests are control engineering, renewable energy systems, remote piloted aircraft systems applications, and engineering education. Throughout his professional life, he has received 24 awards and academic honors. He has conducted 10 Doctoral Theses with eight prizes.