FROM DESIGN TO PROTOTYPING IN THE INTERNET OF THINGS: A DOMOTICS CASE STUDY

Sabrina Sicari[1], Alessandra Rizzardi[1], Alberto Coen-Porisini[1]
[1]Dipartimento di Scienze Teoriche e Applicate, Università degli Studi dell'Insubria, via O. Rossi 9 - 21100 Varese (Italy)

NOTE: Corresponding author: Sabrina Sicari, sabrina.sicari@uninsubria.it

Abstract *– Nowadays, the capability of rapidly designing and prototyping, simple, yet real domotics systems (e.g., smart homes and smart buildings applications) is even more compelling, due to the availability and increasing spread of Internet of Things (IoT) devices. Home automation services enable the remote monitoring of indoor environments and facilities. The main advantages include saving energy consumption and improving the overall management (and users' experience) in certain application domains. The pervasive adoption and diffusion of such remote monitoring solutions is hampered by the timing required for design, prototyping and further developing applications and underlying architecture, which must be often customized on the basis of specific domains' needs and involved entities. To cope with this issue, the paper proposes the analysis and prototyping of a domotics case study, in order to demonstrate the effectiveness of proper IoT-related tools in speeding up the testing phase.*

Keywords – Domotics, Internet of Things, monitoring application, prototyping

1. INTRODUCTION

Applications for indoor and remote monitoring are nowadays adopted in different domains, ranging from smart homes to smart offices, and tailored to many scopes, such as minimizing possible local mismanagement and wastage of resources. Moreover, in order to reduce the negative influences of buildings on the environment, *green* building, which is also known as *sustainable* building, has begun to spread, aimed at at creating a better indoor environmental quality for occupants, while reducing natural resources consumption [1].

Different technologies concur to the realization of remote monitoring applications, which are strictly related to the Internet of Things (IoT) paradigm. They include Wireless Sensor Networks (WSN), Wireless Multimedia Sensor Networks (WMSN), Near Field Communication (NFC), Radio-Frequency Identification (RFID), actuators, and communication protocols such as Message Queue Telemetry Transport (MQTT), ZigBee, Constrained Application Protocol (CoAP), 6LowPAN (IPv6 over Low-Power Wireless Personal Area Networks), and so on [2]. The basic idea behind the IoT paradigm is the possibility of acquiring heterogeneous kinds of information from the environment where IoT devices are placed. Such devices embed both sensing and actuating capabilities, which make them "smart" and enable them to interact with the surrounding environment. Such features allow the IoT system to share a huge amount of information throughout the network and the Internet. Such data can be used to provide customized services to the interested users [3].

To achieve such a goal, the numerous technologies and communication protocols, just mentioned above, should often cooperate, in order to realize an efficient IoT infrastructure and to regulate the information exchange process. Tools, simulators or testing-platforms, for supporting the realization of such IoT infrastructures, from the design towards the development phase, are needed. Their scope is representing all the components acting within the envisioned environment, so as to give an overview of the whole system before real deployment, in a limited scale. Hence, this paper proposes the use of different supporting tools, targeted at the IoT, providing a representative case study related to domotics.

The remainder of this paper is organized as follows. Section 2 investigates the actual state-of-the-art tools and methods used by the researchers for validating remote monitoring systems, thus revealing our motivations. Section 3 presents the technologies and tools adopted for investigating the case study, which is detailed in Section 4. Finally, Section 5 ends the paper, drawing some hints for the direction of future research.

2. RELATED WORKS

A well-investigated field in remote controlling is that of e-health [4] [5], ranging from the monitoring of chronic diseases, to vital signs current status monitoring, and, finally, to the triage prioritization of patients.

Other solutions available concern smart buildings, which mainly include smart homes and smart offices. The work, presented in [6], uses real data-sets, collected from existing smart homes (i.e., reporting information such as energy consumption, lighting, heating, and so on), for testing a middleware conceived to evaluate the security of the information, which is transmitted within the underlying IoT infrastructure. The middleware runs on Raspberry Pi, and it is implemented in Node.js [1], while JSON formal language and the MongoDB [2] database are used for data exchange and storage, respectively.

[1]Node.JS. http://nodejs.org/
[2]MongoDB. http://www.mongodb.org/

Another test bed, consisting of a Raspberry Pi, is detailed in [7]. Also here, the final goal is to evaluate a security protocol for enforcing the usage of control policies. In both cases (i.e., [6] and [7]) the test bed consists of a limited number of devices, thus preventing the conducting of relevant considerations about the scalability of the proposed approaches. Note that such an aspect is not so relevant with respect to the approach presented in this paper.

Instead, the authors of [8] present three different use cases to demonstrate the feasibility and efficiency of their architecture, by measuring home conditions, monitoring home appliances, and controlling home access. Such a solution integrates the IoT paradigm with web services and cloud computing. The following technologies have been adopted for the test bench: Arduino platform for sensing and actuating functionalities; Zigbee for networking; cloud services; JSON data format for information exchange.

A prototype service for a smart office is provided in [9] to evaluate, from a functionalities' viewpoint, the proposed Integrated Semantics Service Platform (ISSP). The solution is based on an ontology and a model for semantic interpretations of user inputs through a proper web app. The whole architecture is based on Mobius [3], which is a *oneM2M-compatible* IoT service platform.

In [10], the work describes a practical realization of an IoT architecture, targeted to the University of Padova (Italy), which allows the interaction of WSN and actuators to standard networks, such as web services. It is an example of smart building, since the IoT network spans the floors and different areas within the Department of Information Engineering. Basic services, such as environmental monitoring and localization, regulated by roles and authorizations, are provided by means of the proposed approach.

A similar work has been carried out at the University of Bari (Italy), where existing hardware and software IoT solutions have been glued together to provide a reliable monitoring system, able to handle both scalar and media data belonging to either Internet Protocol version 4 (IPv4) and IPv6 realms [11]. In more detail, an IoT middleware, named NOS (Networked Smart object) [12], is able to manage IoT heterogeneous data, and has been integrated with: (i) TLSensing platform, which is able to efficiently acquire environmental information; and (ii) an IP camera, in charge of acquiring images from the surrounding environment. An experimental test bed has been been deployed in a university's laboratory, in order to continuously monitor environmental conditions and access control, also against malicious behaviours (e.g., to perform intrusion detection tasks).

Based on a coordinator-based ZigBee network, the smart home control system, presented in [13], has been written as a C# program in charge of simulating the users' behaviour. A similar approach is that of [14], where ZigBee nodes are simulated by means of a well-known WSN simulator, named *NS2*. The main drawback, emerged from such solutions, is that an IoT system is too complex for being simulated by a WSN simulator or by a "simple" software.

Note that, in general, the growth and diffusion of remote monitoring systems was favoured by the availability of sensor devices, able to acquire, in real time, information from the surrounding environment and transmit it throughout the network towards a sink point, which is usually in charge of collecting and processing all the gathered data from a specific application [15]. What emerges from literature is the need for a tool or a set of interoperating tools, able to represent the whole remote monitoring architecture closer, as much as possible, to the future working system, in order to provide provide designers and developers with a complete view of the final architecture and underlying logic, before its real deployment. Such a role has been played by WSN's simulators/emulators for many years [16], but, with the advent of IoT, new systems must be adopted, due to the heterogeneity of the involved devices and to the different services provided. Hence, the main contribution of the work, presented herein, can be summarized as follows:

- The adoption and integration of proper tools and technologies are proposed, in order to represent a domotics IoT scenario.
- A general overview of the envisioned system is given by means of a complete test-bed, to be validated before real deployment on a large scale.

3. TECHNOLOGIES AND TOOLS

Before detailing the case study of interest, the involved technologies and tools are introduced herein. An overview of the envisioned domotics system is provided in Fig. 1, which resembles all the components described for the use case. A demo video is available at *https://youtu.be/-5Gg5I0B3Ak*.

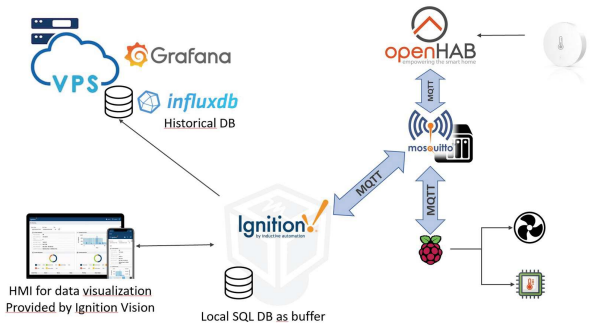

Fig. 1 – Domotics - system architecture

[3] Mobius oneM2M, "oneM2M-compatible IoT service platform". http://wiki.onem2m.org/index.php?title=Open_Source

3.1 MQTT

MQTT[4] is a publish&subscribe network broker-based messaging protocol, which is used to transport messages between devices in IoT networks or, in general, in constrained scenarios. Note that it is largely used in the IoT domain thanks to its robustness and power-saving communication. MQTT is based on two types of entities: message broker and clients. In more detail, a one-to-many message distribution is performed as well as another feature involved in the decoupling of information between sources and consumers. In general, MQTT is agnostic about the content of the payload. The broker can be implemented by using Mosquitto [5], but other solutions are also available, such as HiveMQ [6].

The concept of *topic* is fundamental in MQTT; it consists of strings used by the broker to filter messages gathered by the connected client; a topic has one or more levels, separated by a forward slash, so as to obtain a logical tree structure. Topics are used by clients for publishing messages and for subscribing to the updates from other clients, thus avoiding a continuous polling among producers and consumers. There is the possibility to subscribe to an exact topic or to multiple topics at once by using the wildcards, represented by the following symbols: (i) + for a single-level wildcard (i.e., exactly one topic level); (ii) # for a multi-level wildcard (i.e., an arbitrary number of topic levels). When a message is published under a certain topic, it is delivered to each matching subscription registered at that time.

In the case study presented in Section 4, MQTT protocol plays a central role in message passing, due to its efficiency for the investigated scenario.

3.2 InluxDBd

InluxDBd[7] is a database belonging to the NoSQL family. It has been specifically designed and developed for managing time-series data, thus making it an ideal choice for periodically logging sensor information. Data is stored into "measurements" by using timestamps, fields and tags. Fields are used to store data information, which can be strings, floats, integer or boolean and are always associated with a timestamp. Tags are similar to fields, but are also indexed; this allows the storage of important metadata in tags to optimize querying performance. InluxDBd is adopted in the case study, presented in Section 4, since its data structure perfectly fits the need of heterogeneity, which is dictated by IoT environments.

[4]MQTT v3.1/v3.1.1, https://mqtt.org/mqtt-specification/
[5]Mosquitto, open source MQTT v3.1/v3.1.1 broker. http://mosquitto.org
[6]HiveMQ MQTT broker. https://www.hivemq.com
[7]InfluxDB, time series platform. https://www.influxdata.com/

3.3 OpenHAB

Open Home Automation Bus (OpenHAB) [8] is an open-source project home application platform used to run smart homes. It naively supports many devices and allows the user to further extend its capabilities by installing modules and plugins. Moreover, OpenHAB allows the writing of custom logical rules, which can be triggered using deployed sensors and perform user-defined actions (e.g., turn on lights on a given time or when a motion sensor is activated). OpenHAB makes use of various different concepts to model the smart home environment, two of which are relevant for the case study presented in this paper:

- *Things* can be seen as entities that can be physically added to the system, like a vendor sensor gateway, or virtual, like a web service which can provide information to the system
- *Items*, instead, represent functionality used by the application. For example, the temperature values read by a sensor, or the current state of a switch, are considered as *items*; while the sensor itself is the *things* which provides such *items*.

For logical operations, OpenHAB uses *rules*, which are composed in the following way:

- Rule's *name*, which defines a unique name to reference the rule
- *When* statement, which provides the trigger to activate the rule
- *Then* statement, which defines the tasks to be performed when the rule is triggered.

3.4 Ignition

Ignition [9] is a commercial, server-based cross-platform software, which is managed through web technology. Built with customization in mind, it supports a modular structure so that its deployment can be tailored for every specific use case. Ignition includes a Human Machine Interface / Supervisory Control And Data Acquisition (HMI/SCADA), which can be built, in a customized way, depending on the intended purpose. Ignition makes use of *tags* as points of data; these can be both static or dynamic on the basis of the final scope. While it provides a set of predefined types, it also allows the user to extend them through the use of *User Data Types (UDT tags)*. The basic *tags* are the following:

- *OPC Tags* are particular kinds of tag, which use the Open Process Connectivity (OPC) standard to communicate and read/write values directly to the Programmable Logic Controller (PLC).

[8]Openhab, open source automation software. https://www.openhab.org
[9]Ignition software. https://inductiveautomation.com

- *Memory Tags* are tags which hold and store information; they can be seen as variables in a programming language.
- *Expression Tags* are tags which are driven by a user-defined expression, such as a mathematical operation, a logical operation, and so on.
- *Query Tags* are tags which pool their value from an SQL statement. They can also refer to other tags to build dynamic queries.
- *Reference Tags* are tags which simply refer to other tags to fetch their value.

Ignition, and the next Grafana tool, will be coupled with OpenHAB to realize a simple yet real domotics system with the support of real devices, as presented in Section 4.

3.5 Grafana

Grafana [10] is an open-source web-based tool for data visualization and analysis. It allows the modeling of custom dashboards, based on the required use cases and supports different data sources, like: InfluxDB, Microsoft SQL Server, PostgresSQL, AWS CloudWatch, etc. Also, it allows the development and installation of custom modules/plugins which can expand its capabilities.

4. CASE STUDY AND PROTOTYPE

In the case study presented herein, real devices are directly connected to software applications and tools, which are able to change their status. The idea is to build a system using the Ignition software SCADA solution to control real devices connected to it. The list of used devices includes: (i) a RaspberryPi Zero W[11] with an attached computer fan; (ii) three Xiaomi room temperature and humidity sensors; and (iii) two Xiaomi smart plugs. Also, different host systems are involved: (i) Virtualbox to host the SCADA system; (ii) home NAS (Network Attached Storage) to host the MQTT broker (deployed using Mosquitto); and (iii) cloud VPS to host historical database and data visualization UI.
The RaspberryPi Zero W behaves like an IoT enabled PLC. PLC devices are equipped with sensors (to gather information) and actuators (to perform actions). In this case, CPU temperature information is used to feed the sensor's data. For the actuator, instead, an external fan is used and controlled through the GPIO pins using Pulse Width Modulation (PWM). PWM is a method used to control devices that require power or electricity. It essentially makes use of a digital signal, which is periodically turned on or off to modulate the connected device. In particular, PWM is used to control the fans' motor; the larger the time frame between pulses is, the slower the motor turns. The envisioned scheme is shown in Fig. 2.

[10]Grafana, interactive visualization tool. https://grafana.com
[11]https://www.raspberrypi.org/products/raspberry-pi-zero-w/

The RaspberryPi communication is managed through the usage of MQTT, where:

- Sensors' information is published to the topic *rpi/cpu/temperature*
- Actuators' information is fetched by subscribing to the topic *rpi/fan/speed*

All the required logic is written using Python and executed at boot time by means of a *crontab task*.

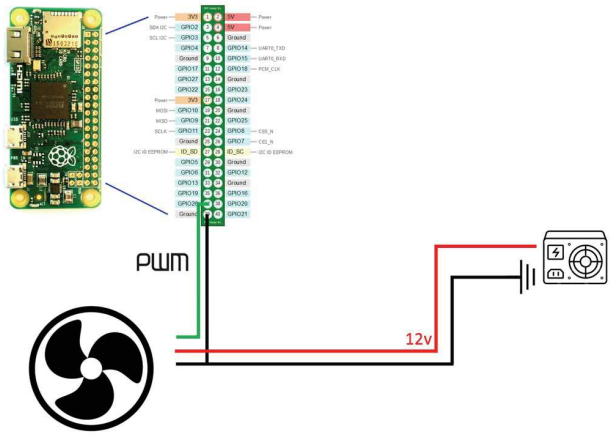

Fig. 2 – Domotics - fan's connection schema

OpenHAB, presented in Section 3 and hosted on the home NAS, is used to bridge the connection among the proprietary Xiaomi smart home sensors. The adopted modules are: (i) Xiaomi Smart Home Binding[12], which is used to communicate with the sensor gateway; and (ii) MQTT Binding[13], which is used to connect to an MQTT broker. Configuration for the Xiaomi binding is done through the OpenHAB web GUI, by providing the gateway IP and private API key, as shown in Fig. 3. By means of a web GUI, it is possible to automatically search and add all detected/connected sensors creating for each a dedicated *item*. Such an *item* is used in *rule*'s definition to identify the sensor. The MQTT binding is configured by adding an appropriate *thing* file in the OpenHAB configuration folder, thus allowing it to be referred in MQTT communication. The next step is defining the logical rules to follow for publishing sensor data over MQTT. These are defined by adding *rules*' file into the OpenHAB configuration folder. The rule triggers on every detected state change on the observed sensor by publishing its new value to the appropriate topic, as defined before.
The core of the system is the Ignition SCADA, which is the software in charge of reading and managing all attached devices, connecting via the MQTT protocol. Ignition leans on a Microsoft SQL server database, which is used as a local buffer to periodically store and change the status of sensors' information in case the connection towards the historical server is lost, but also to manage data migration

[12]https://www.openhab.org/addons/bindings/mihome/
[13]https://www.openhab.org/addons/bindings/mqtt/

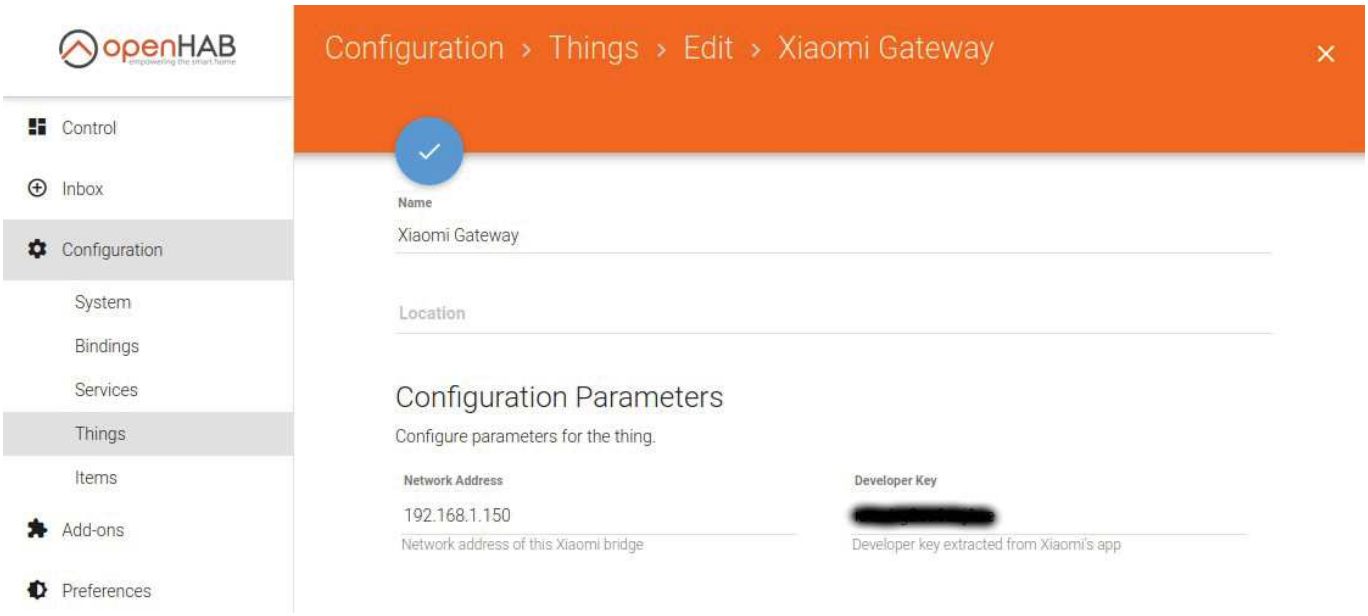

Fig. 3 – Domotics - OpenHAB Xiaomi binding configuration

towards the historical database, which is hosted on an external VPS server. The database structure is very simple, since it includes the two tables shown in Fig. 4. Instead, the topics' structure is the following:

- RaspberryPi CPU temperature *rpi/cpu/temperature*
- Current fan speed *rpi/fan/rpm*
- Room's temperature and humidity sensors:
  - *home/room_1/temperature* & *home/room_1/humidity*
  - *home/room_2/temperature* & *home/room_2/humidity*
  - *home/kitchen/temperature* & *home/kitchen/humidity*
- Current light state:
  - *home/light_1/state*
  - *home/light_2/state*

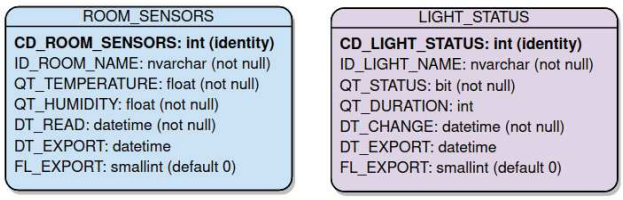

Fig. 4 – Domotics - table structure

A memory tag is added to hold the current desired fan speed, ranging from 0-100. Such a tag implements a Python script, which triggers on a value update, checks if the new value is valid and publishes it to the dedicated topic *rpi/fan/speed*. To automatically manage the fan, two support memory tags are needed: (i) a boolean, to toggle on or off the automatic fan profile; and (ii) a float, to set the desired target CPU temperature. Also, a script is applied to the CPU temperature tag, which triggers on each new read value, comparing it to the desired temperature and deciding if its necessary to turn the fan on or off. To avoid continuously switching the fan state, due to the temperature's fluctuations around the threshold value, the fan is turned off once a temperature, which is 2°C lower than the target one, is reached. A scheme of the just described behaviour is sketched in Fig. 5.

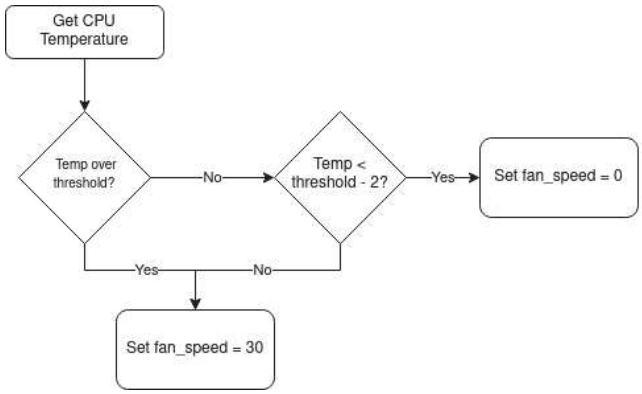

Fig. 5 – Domotics - fan's profile flowchart

Furthermore, a web GUI is needed to visualize and control the connected devices; it is realized by means of Vision, which is an UI building tool, provided by Ignition, and it is able to build a local HMI, as shown in Fig. 6. Vision is composed of four parts:

- MQTT broker status, where the right LED indicates if the broker is currently available (green) or not (red)

- Room temperature and humidity, which indicate, for each room, the last detected information, while also displaying a chart with the trend of the last hours. The chart toggles between temperature and humidity by pressing the appropriate current value
- Light status, which shows the current light state and the last date-time when it toggled
- RaspberryPi fan control, which displays a gauge with the current real-time CPU temperature. This is divided into three colour-coded areas: (i) green as desired temperature; (ii) yellow above target; and (iii) red as over temperature. The user can interact with the dial being able to change the desired temperature. Also, it is possible to override the automatic fan controller with the appropriate switch and, then, use the slider to manually set the fan speed.

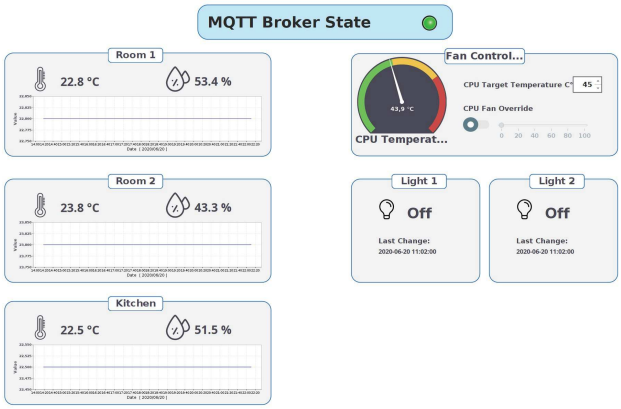

Fig. 6 – Domotics - Vision HMI

As just anticipated, the historical database is hosted on an external VPS server, and it is paired with a data visualization tool, named Grafana (see Section 3), that allows the visualization of the gathered (and stored) data in various user's defined charts. The database used for this task is InluxDBd(see Section 3) mainly for two reasons: (i) it is designed specifically to manage time-based information; and (ii) it is officially supported by the Grafana suite. For security reasons, two separate accounts have been defined on the historical database: (i) read/write, for the migration task; and (ii) read only, for the Grafana connection. As stated in Section 3, InluxDBddefines two types of data: tags and fields. For this specific use case, the $roomID$ and $lightID$ are added as tags, since they are mainly used in the $WHERE$ statement to filter data; while all the other information is stored into fields. Two measurements are used: one for rooms and one for lights. Hence, as shown in Fig. 7, two types of panels are visible: (i) a chart to plot the temperature/humidity trend; and (ii) a panel to visualize how long the light states lasted.

5. CONCLUSION

Early design and prototyping are fundamental to speed up the development process of monitoring systems. To achieve such a goal, proper tools and technologies must be adopted. In this respect, a domotics case study has been pointed out in this paper, where different technologies, protocols and languages are grouped together without worrying about interoperability issues, thanks to the capabilities of the adopted tools to interact between themselves. In that sense, the presented approach represents a viable solution for performing preliminary tests on a domotics IoT-based scenario. Note that other technologies could be adopted for the same purpose, such as MongoDB as data store, instead of InfluxDB, and CoAP (Constrained Application Protocol) as transmission protocol, instead of MQTT. Both such solutions are targeted to IoT and constrained scenarios; however MongoDB is a document-oriented (and not time based, as InfluxDB) database, while CoAP does not follow a publish and subscribe philosophy, as MQTT. For such reasons, in this work InluxDBdand MQTT have been preferred, since they better fit the requirements of a domotics context, where handling data following a topics' hierarchy, instead of unstructured information, represents the most viable solution.

Two important aspects still deserve attention, as an open research activity: scalability and securit and privacy. Such a kind of analysis could be carried out by defining re-usable modules and components to be integrated (and replicated) in a more complex system. Instead, security&privacy requirements can be achieved at various levels: ranging from securing the MQTT communication exchange [17] [18], to protecting the data when they are stored into the database [19] [20], or to providing security and privacy policy enforcement mechanisms at the IoT core platform's level [21].

Fig. 7 – Domotics - Grafana user's defined dashboard

REFERENCES

[1] Yang Geng, Wenjie Ji, Zhe Wang, Borong Lin, and Yingxin Zhu. "A review of operating performance in green buildings: Energy use, indoor environmental quality and occupant satisfaction". In: *Energy and Buildings* 183 (2019), pp. 500–514.

[2] Shadi Al-Sarawi, Mohammed Anbar, Kamal Alieyan, and Mahmood Alzubaidi. "Internet of Things (IoT) communication protocols". In: *2017 8th International conference on information technology (ICIT)*. IEEE. 2017, pp. 685–690.

[3] Maria Stoyanova, Yannis Nikoloudakis, Spyridon Panagiotakis, Evangelos Pallis, and Evangelos K Markakis. "A survey on the internet of things (IoT) forensics: challenges, approaches, and open issues". In: *IEEE Communications Surveys & Tutorials* 22.2 (2020), pp. 1191–1221.

[4] OS Albahri, AS Albahri, KI Mohammed, AA Zaidan, BB Zaidan, M Hashim, and Omar H Salman. "Systematic review of real-time remote health monitoring system in triage and priority-based sensor technology: Taxonomy, open challenges, motivation and recommendations". In: *Journal of medical systems* 42.5 (2018), p. 80.

[5] Rachael C Walker, Allison Tong, Kirsten Howard, and Suetonia C Palmer. "Patient expectations and experiences of remote monitoring for chronic diseases: systematic review and thematic synthesis of qualitative studies". In: *International journal of medical informatics* 124 (2019), pp. 78–85.

[6] Sabrina Sicari, Alessandra Rizzardi, Daniele Miorandi, and Alberto Coen-Porisini. "Securing the smart home: A real case study". In: *Internet Technology Letters* 1.3 (2018), e22.

[7] Antonio La Marra, Fabio Martinelli, Paolo Mori, and Andrea Saracino. "Implementing usage control in internet of things: a smart home use case". In: *Trustcom/BigDataSE/ICESS, 2017 IEEE*. IEEE. 2017, pp. 1056–1063.

[8] Moataz Soliman, Tobi Abiodun, Tarek Hamouda, Jiehan Zhou, and Chung-Horng Lung. "Smart home: Integrating internet of things with web services and cloud computing". In: *2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE. 2013, pp. 317–320.

[9] Minwoo Ryu, Jaeho Kim, and Jaeseok Yun. "Integrated semantics service platform for the Internet of Things: A case study of a smart office". In: *Sensors* 15.1 (2015), pp. 2137–2160.

[10] Angelo P Castellani, Nicola Bui, Paolo Casari, Michele Rossi, Zach Shelby, and Michele Zorzi. "Architecture and protocols for the internet of things: A case study". In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on.* IEEE. 2010, pp. 678–683.

[11] D. Costantino, G. Malagnini, F. Carrera, A. Rizzardi, P. Boccadoro, S. Sicari, and L. A. Grieco. "Solving Interoperability within the Smart Building: A Real Test-Bed". In: *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*. May 2018, pp. 1–6.

[12] S.Sicari, A. Rizzardi, D. Miorandi, C. Cappiello, and A. Coen-Porisini. "A secure and quality-aware prototypical architecture for the Internet of Things". In: *Information Systems* 58 (2016), pp. 43–55.

[13] Murad Khan, Bhagya Nathali Silva, and Kijun Han. "Internet of things based energy aware smart home control system". In: *IEEE Access* 4 (2016), pp. 7556–7566.

[14] P Xiang. "Design of smart home system based on the technology of internet of things". In: *Research Journal of Applied Sciences, Engineering and Technology* 4.14 (2012), pp. 2236–2240.

[15] SR Jino Ramson and D Jackuline Moni. "Applications of wireless sensor networks - A survey". In: *2017 international conference on innovations in electrical, electronics, instrumentation and media technology (ICEEIMT)*. IEEE. 2017, pp. 325–329.

[16] Ivan Minakov, Roberto Passerone, Alessandra Rizzardi, and Sabrina Sicari. "A comparative study of recent wireless sensor network simulators". In: *ACM Transactions on Sensor Networks (TOSN)* 12.3 (2016), p. 20.

[17] A. Rizzardi, S. Sicari, D. Miorandi, and A. Coen-Porisini. "AUPS: An Open Source AUthenticated Publish/Subscribe system for the Internet of Things". In: *Information Systems* 62 (2016), pp. 29–41.

[18] Chang-Seop Park and Hye-Min Nam. "Security Architecture and Protocols for Secure MQTT-SN". In: *IEEE Access* 8 (2020), pp. 226422–226436.

[19] Guo Yubin, Zhang Liankuan, Lin Fengren, and Li Ximing. "A solution for privacy-preserving data manipulation and query on NoSQL database". In: *Journal of Computers* 8.6 (2013), pp. 1427–1432.

[20] Kanika Goel and Arthur HM Ter Hofstede. "Privacy-Breaching Patterns in NoSQL Databases". In: *IEEE Access* 9 (2021), pp. 35229–35239.

[21] S. Sicari, A. Rizzardi, D. Miorandi, C.Cappiello, and A. Coen-Porisini. "Security Policy Enforcement for Networked Smart Objects". In: *Computer Networks* 108 (2016), pp. 133–147.

AUTHORS

Sabrina Sicari is Associate Professor at University of Insubria (Varese). She received a degree in Electronical Engineering, 110/110 cum laude, from University of Catania, in 2002, where in 2006 she got a Ph.D. in Computer and Telecommunications Engineering. She is a member of COMNET, IEEE IoT, ETT, ITL editorial board. Her research activity focuses on security, privacy and trust in WSN, WMSN, IoT, and distributed systems.

Alessandra Rizzardi received a BS/MS degree in Computer Science 110/110 cum laude at University of Insubria (Varese), in 2011/2013. In 2016 she got a Ph.D. in Computer Science and Computational Mathematics at the same university, under the guidance of Prof. Sabrina Sicari. She is Assistant Professor in Software Engineering at the University of Insubria. Her research activity is on WSN and IoT security issues.

Alberto Coen-Porisini received his Dr. Eng. degree and Ph.D. in Computer Engineering from Politecnico di Milano in 1987 and 1992, respectively. He has been Professor of Software Engineering at Università degli Studi dell'Insubria since 2001, Dean of the School of Science from 2006 and Dean since 2012. His research regards specification/design of real-time systems, privacy models and WSN.