Advances in Science, Technology and Engineering Systems Journal Vol. 9, No. 4, 14-20 (2024)

www.astesj.com

ASTESJ

ISSN: 2415-6698

From Sensors to Data: Model and Architecture of an IoT Public Network

Stefania Nanni[1], Massimo Carboni[1], Gianluca Mazzini[2]

[1] *LepidaScpa, Research & Prototypes, Bologna, 40128, Italy*

[2] *LepidaScpa, CEO, Bologna, 40128, Italy*

ARTICLE INFO

Article history:
Received: 03 April, 2024
Revised: 28 May,2024
Accepted: 29 June, 2024
Online: 11 July, 2024

Keywords:
IoT network
Decoder
Data extraction

ABSTRACT

RetePAIoT of Emilia-Romagna region is an IoT Public Network, financed by Emilia-Romagna Region and developed by Lepida Scpa, where citizens, private companies and Public Administrations can integrate free of charge their own sensors of any type and anywhere in the region. The main objective of the project is to provide a facility to implement the IoT paradigm, receiving data potentially from thousands of different sensors from the territory and to make them available to their owners and, in aggregate or anonymous form, to Public Administrations for their institutional purposes. In this context the interpretation of payloads sent by sensors, i.e. the extraction of the values measured by sensors, as well sharing them with all authorized subjects. are fundamental aspects that present a significant complexity due to the variety and unplannable context of the project. This paper illustrates the model and the architecture of a solution for the automatic extraction of values potentially coming from thousands of different sensors, which therefore requires a high level of flexibility, robustness and scalability as well as different methods for sharing them with third parties, depending on purposes and technical level required.

1. Introduction

As illustrated in the original paper [1], in 2019 LepidaScpa launched a project, financed by Emilia-Romagna Region, called retePAIoT, aimed at covering the entire regional territory with a LoRaWan network [2], offering an IoT infrastructure to both Public Administrations and to private citizens or companies to connect their own sensors.

Several actors are participating to this project, with different roles:

- LepidaScpA is in charge of the development and the maintenance of the IoT infrastructure: gateways, server network, included the web platform for the management of sensors, their data and users;
- Municipalities have to provide free of charge places of installation of gateways and to advertise the initiative on their own territory;
- Municipalities have to provide free of charge places of installation of gateways and to advertise the initiative on their own territory;
- End users (both public and private) have to purchase sensors. Type and model of sensors can be any, but users have to provide LepidaScpa with information on those not yet present in the catalog, to be update. Users, in fact, have also to register their own sensors into retePAIoT network, through retePAIoT web interface [3], specifying the correct brand and model of sensors, from which decoding rules of payload of sensors messages depend.

The main objective of the retePAIoT project is to make available a free of charge IoT regional network to Public Administrations as well as private users or companies, to facilitate the installation of their own sensors, collecting data from the territory and making them available to owners of the sensors and to Public Administrations for their institutional purposes [4]. This means that the measurements sent by the sensors integrated in the

* Corresponding Author: Stefania Nanni, Lepida ScpA; stefania.nanni@lepida.it

https://dx.doi.org/10.25046/aj090403

public IoT network, retePAIoT, can be consulted both by owners of the sensors, but also, anonymously or in aggregate form, by the Public Administrations interested in exploiting the data coming from the regional territory. This important goal is possible only if the retePAIoT project, in addition to facilitate the installation and the collection of data sent by the sensors, also provides their decoding, i.e. the extraction of the measured values, making them usable, through appropriate interfaces, both to the owners of the sensors and to the Public Administrations.

To encourage the use of the retePAIoT project by all interested public and private users, no limits have been set to the types, models and brands of sensors that can be used. For this reason, the catalogue of the sensors that can be registered in retePAIoT is constantly updated upon users request with new models and types.

It is important to underline that other LoRaWan networks exist in Europe and around the world which are public and free and which allow different types of sensors to be integrated by different users. An exemplary case is the TTN network, The Thing Network [5], which, in addition to being widespread in Europe, especially in France, and in the world, constitutes a global network for the integration of both sensors and LoRaWan gateways. But while both networks, TTN and retePAIoT, share the primary purpose of providing a network infrastructure that is accessible to anyone at no cost, promoting innovation and widespread adoption of IoT technology, retePAIoT has the additional and fundamental objective to use the data collected by the sensors.

This purpose is not only peculiar to the retePAIoT network, compared to other public ones, which normally delegate it to single users, but it involves a series of onerous activities such as knowledge, cataloguing, validation, description of the measurements and the corresponding units within the Data Base of all integrated sensors, as well as the implementation of a new architecture for the automatic extraction of data from the messages sent by the sensors, which constitutes the real added value and aspect innovative of the solution presented in the original paper [1] and in this one.

In this paper it will be highlighted the importance, the critical issues and the automatic solution adopted for the decoding of the payloads and for the extraction of the data within a complex scenario such as an IoT public network like retePAIoT and some different methods and interfaces to share them to different users, according to their needs.

For this purpose, this paper starts form a brief overview of the state of the art of some useful management features offered by ChirpStack [6], that is the open LoRaWan server network used to manage retePAIoT network, and on which the model and architecture of the solution proposed in this paper is based. The rest of this paper is organised as follows: the third section briefly illustrates the main architecture of the Internet of Things public network, retePAIoT; section IV describes the structure of the centralized database, with particular reference to the fundamental extension of sensors registry tables introduced to resolve criticalities and to guarantee the requirements of payload decoding service; section V focuses on the new architecture implemented to automatically decode payloads sent by sensors, based on a new feature made available by ChirpStack, its logical flow, closely based on sensors database model extended, and the relevant goals and advantages that it achieves respect the original one [1]; section VI describes different interfaces made available by retePAIoT network to share data to different users and objectives; section VII describes a significant use data case, and the last one summarises the main results achieved by retePAIoT network.

2. The State of the Art

ChirpStack is an open-source platform for managing and monitoring LoRaWan networks. It provides a modular software suite that enables network operators, developers and end users to build and manage scalable and reliable LoRaWan networks. ChirpStack Application Server is a module of ChirpStack that manages IoT applications, allowing developers to create and manage custom applications to analyze and interact with data from LoRaWan devices. In particular, the 'device profile' and the 'application' are two entities, managed by the Application server module, which respectively allow to specify the communication and configuration settings of the devices and to manage a specific IoT application or use case within the ChirpStack platform.

Starting from ChirpStack version 3.0, decoding rules of payloads devices have been associated to the 'device profile' entity instead of the ‘application’ one, making it possible to avoid duplication of decoding rules in the very common cases in which devices of the same type were used in applications associated to different users.

The new positioning of the payload decoding function constitutes the opportunity to provide the decoding of the payloads of the various devices directly within the ChirpStack, in place of an external module, using directly the javascript code, normally provided by the devices manufacturers, instead of requiring the development of different software modules, as implemented in the first release of retePAIoT [1]. ‘Application' is now only a logical entity that allows network operators to create and to manage customized IoT applications according to the specific needs of different use cases and users, relatively, for example, to different modalities to share data of devices associated with other external systems. The two entities ‘device profile’ and ‘application’ managed by ChripStack in last releases are at the base of improved model and architecture of retePAIoT platform, for the extraction and the sharing of the data, that are two fundamental aspects for the use of sensors data, which constitutes the final objective of retePAIoT and, in general of all IoT networks.

3. retePAIoT Network Architecture

The basic architecture of a public network for the internet of things like retePAIoT is shown in Fig.1, where main components, sensors, gateways and network server, and types of connections are highlighted: the black line represents an internet connection, the violet line a Lepida fiber connection, the dotted line a communication based on wireless LoraWan protocol, whose specifications can be found in [7].

One or more LoraWan gateway are installed in each participating municipality, and represent the meeting point between two types of communication: on the one hand, through the LoraWAN protocol, they receive the data coming from the users sensors, on the other they are connected to one of the fiber

points of access of the Lepida Network, allowing the sensors messages to reach the LoraWan server.

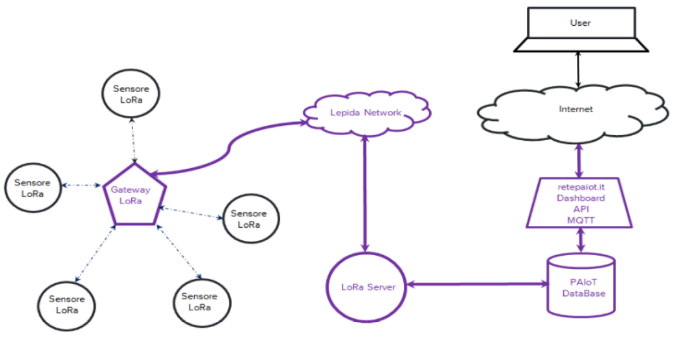

Figure 1: Architecture of internet of things public network, retePAIoT

This server is a virtual machine hosted in one of the four Regional Data Centers managed by LepidaScpA [8], and it undertakes the registration of sensors in the network and the subsequent establishment of encrypted sessions through which payloads are transmitted. The software installed for the management of the protocol is the open source LoRa Server ChirpStack [6].

Every user must register himself and the sensors, for which he is responsible or owner, through the web portal, another virtual machine in the same Datacenter where the LoRaServer is hosted.

The lack of a communication standard shared between all IoT devices is a problem whose solution strongly depends on the technology and the scope in which you are working [9], [10], [11]. The major criticality addressed in this paper arises from the need to postpone the decoding process with respect to that of messages reception, in order to better comply with the nature of the retePAIoT network which is free, open to any brand and model that users may require and mindful of the decoupling between raw and processed information.

In the following paragraphs, the architectural and functional aspects to save sensors payloads and to interpret and extract their data and different methods to share them will be examined in depth, because, as already highlighted, they constitute essential services of the project that present some elements of complexity which require not only a critical and in-depth analysis, but it also deserves a specific focus, that this paper intends to highlight, together with the solution devised and implemented to resolve them.

4. Extended DataBase

The database of retePAIoT network is an Oracle relational database that allows to manage both sensors and users data and to store the payloads and the values of the measured quantities, as shown in Figure 2, as already illustrated in a previous paper [3].

The next one [1], instead, focused on extension of some tables of the database provided for the description of the sensors, 'sensor', and for saving of their messages, ‘sensor-value’, which became necessary to deal with some cricities related to the decoding process. In this paragraph it is considered useful to summarise the critical aspects that made it necessary and above all the advantages that derived from it.

The problems that managers of retePAIoT network have to face and have to solve in retePAIoT scenario are basically three:

1. check of the correctness of the brand and the model specified by the user for a sensor, through the comparison between the format of expected payload and that of payloads actually received
2. description of all measures, provided by a sensor, in ‘sensor-measure’ table, according to sensor brand and model
3. implementation of the rules for extracting and storing the values from payloads for every new type, brand a model of a sensor

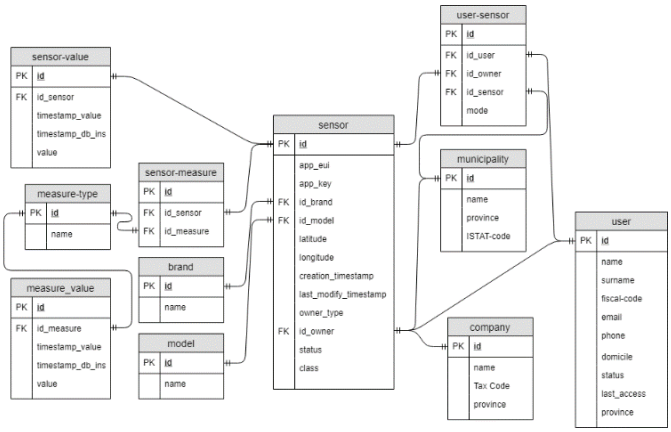

Figure 2: DataBase schema of retePAIoT

These problems imply that it is not possible to ensure that a sensor has been completely described within the database, through the specification of the corresponding measurements neither the payload decoding service is active at the very moment in which its payloads start to arrive.

To cope with these problems and guarantee the correct decoding of the payloads at any later time without any loss of information, an extension of two tables, 'sensor' and 'sensor-value' tables, which respectively contain the details of all the sensors registered on the PAIoT network and all payloads received by sensors, was provided:

‘sensor’ table:

- instance (integer, default = 0)
- interpretation (boolean, default = false)

‘sensor-values’ table:

- interpreted (boolean, default = false)

The 'sensor.instance= 0' field indicates that the correctness of the model and brand specified by the user for the sensor has not yet been verified and that, in particular, within the database, in the 'sensor-measure' table the sensor corresponding measures have not been described yet.

The 'sensor.interpretation = false' field indicates that the sensor payloads cannot be decoded yet, either because the sensor has not yet been instantiated, 'sensor.istance = 0', or because the decoding rules of the specific sensor have not been implemented yet.

The sensor-value.interpreted=false' field keeps track of the payloads already received from the network server and stored within the database in the 'sensor-value' table, but not interpreted yet.

The extension of the 'sensor.instance' field allows to complete the description of the corresponding measurements of a sensor at any time after the registration and only thereafter the validation of the corresponding model.

The extension of the 'sensor.interpretation' field allows to postpone the decoding of the payloads until all the conditions that make it possible are verified:

- consistent description of the sensor corresponding measures within the database in the 'measure' table, corresponding to 'sensor.istance' = 2
- availability of the rules for extracting the corresponding values in the decoding module 'sensor.interpretation = true'

Finally, the extension of the 'sensor-values' table with the 'interpreted' field allows to keep track of the received and stored payloads, which have not yet been decoded and to be able to do that at any subsequent time, without losing any information.

5. Decoding Modules Improvement

As already explained in the original paper [1], in retePAIoT platform the process of receiving, decrypting, saving and interpreting messages sent by sensors via the LoRaWan network is carried out by ChirpStack network server and two main modules: 'archiver payload' and 'decoder'. In the original release [1], however, the 'archiver payload' module had only the task of saving the messages decrypted by the network server into 'sensor-value' table, completely delegating the extraction of the corresponding values to the asynchronous 'decoder' module, when all the conditions necessary for decoding payloads i.e. description of the sensor measurements within the database (sensor.istance = 'true') and the implementation of the payload decoding rules (sensor.interpretation='true'), had been verified. The need to provide an asynchronous 'decoder' module capable of decoding payloads in a phase following the reception of the messages, is intrinsic into the complex and unplannable scenario in which the public network RetePaIoT operates and has been also maintained in the advanced solution illustrated in this paper. The possibility offered by the new versions of ChirpStack of inserting the payload decoding code into the ChirpStack 'device_profile' for each different type of sensor has the big advantage of exploiting the javascript code normally provided by the sensors manufacturers, optimising its development and maintenance.

The advanced solution presented in this paper provides the possibility of decoding the payloads of the messages both in synchronous mode by the ChirpStack server, once the configuration of the new sensors in retePAIoT is fully operational, and in asynchronous mode, as a robust and flexible mechanism for recovering all the messages received and not yet decoded or even for the re-execution of the message decoding phase in case of need to correct any decoding errors.

It is important to underline that, for each type of sensor, the advanced solution presented in this paper uses the same decoding code implemented in the correspondent 'device profile' both in the synchronous and asynchronous mode. To this end, the asynchronous 'decoder' module has been modified so that, for each record in the 'sensor-value' table to be decoded, it recalls the decoder code foreseen for the 'device_profile' assigned to the device sending the message. In the second release, therefore, the asynchronous decoding of the messages process has been improved regarding the following two aspects:

- entrusting the network server with the real-time decoding of the messages coming from the sensors already associated with a profile and already validated and described (sensor.instance = 2 and sensor.interpretation = true)
- taking charge of the decoding only of those messages not still interpreted by the network server ('sensor-value.interpreted = 'false'), but using the same javascript code specified in the ChirpStack devices profiles, once it has been made available

In the new release of retePAIoT the process of managing messages received by ChirpStack network server works as follow (Fig.3):

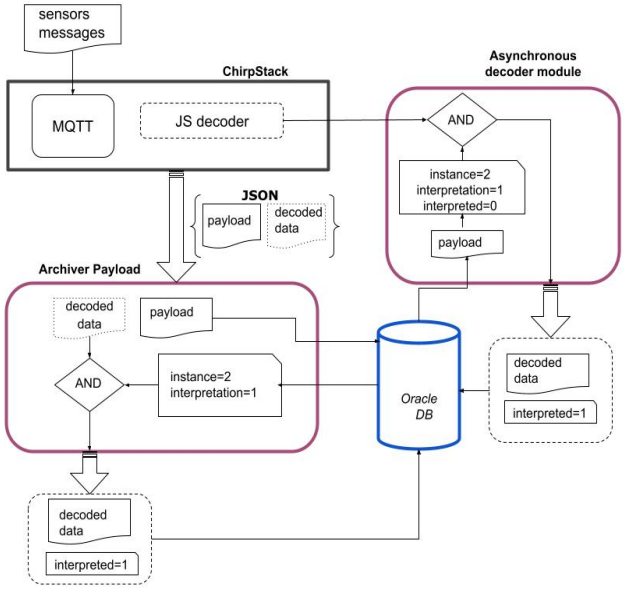

Figure 3: Decoding improved modules data flow

5.1. "archiver payload" module

This module, always running, via the mqtt protocol performs a subscribe on the queue with topic "application" of the LoRaWan ChirpStack network server.

When the network server receives a frame from a device registered on its network database, it decrypts the message, it packs all the information of the frame, both the payload and transmission metadata (i.e. SF, RSSI, SNR, ect.) into a json format and it publishes them on the internal mqtt server. If for the device, that has sent the message, is associated a 'profile device' for which is provided also a decoder java script code, that knows how to analyse, parse and extract the values from the hexadecimal data of the payload, the server network also extracts the values contained in the payload, adding them to the published json data.

The callback function registered for the 'onMessage' event of the topic 'application' is invoked in the "archiver module", that receives the json of the message just published from the server network as an argument and it provides to save the content, i.e. payload and metadata transmission of the message in 'sensor-value' table of the database and, if present and sensor.istance = 2 and sensor.interpretation=true, also moves the decoded data, in 'measure_value' table. In this case the field 'sensor-value.interpreted' of the record just saved in table 'sensor-value' is set to 'true', meaning that the payload has already been interpreted, otherwise to 'false'.

5.2. "decoder" module

'Decoder' module is an asynchronous module that queries the database by joining the 'sensor' and 'sensor-value' tables, in order to retrieve all and only payloads which still need to be interpreted, for which the decoder is now available and whose measurements are now described within the database, that satisfy, therefore, the following correspondents conditions, expressed in the ' where' clause:

- sensor-value.interpreted=0
- sensor.interpretation=1
- sensor.istance=2

It should be highlighted that, in case of crash of the 'decoder' module, the 'sensor-value.interpreted' flag remains set to zero: this does not involve loss of decoded data, since the payloads can always be reprocessed asynchronously at any later time. The same applies in case one of the above conditions (instance, interpretation) is not satisfied at the moment of first processing but changes afterwards. Additionally it becomes possible to reprocess the payloads as many many times as needed in case of errors or updates in the decoder javascript.

5.3. The transition to the new architecture

The design of the data flow in steps controlled by flags set in the database for each sensor and payload was crucial also to allow for a seamless transition from the previous to the new system architecture. With the previous design, a php decoder script was always alert to catch newly stored payloads into the database with 'sensor-value.interpreted=false' flag. This software was not terminated but is being kept running to process the data from those sensors that do not have a decoder profiled in the ChirpStack server yet. At the same time, those payloads arriving from sensors with a profile equipped with a decoder javascript are given a 'sensor-value.interpreted=true' flag before being written to the database in such a way that the php script of the old architecture will not catch them up. Conversely, the payloads that have not been decoded directly by the network server because no code was added to their sensor profile, are given an 'sensor-value.interpreted=0' flag allowing the payload module to store the raw payload in the database without attempting to recover the decoded information and leaving to the php script the decoding task.

The two systems, old and new architecture, are therefore running at the same time: the old php decoders are progressively dismissed while new js-decoders are added to the ChirpStack sensor profiles. No sudden switch from the old to the new system was necessary, allowing the administrators to test the new decoders with ease one at the time.

5.4. Derived measures

It is often useful to compute derived measurements from the data that has been decoded, for example changes in the unit of measure or collection of cumulated values. If a description of the new measures is provided for a sensor in the 'sensor-measure' table, the relative information is accessed during the processing of the payload and a new attribute 'sensor-values.to_be_derived' can be set for the payload and stored in the database. This technique allows to compute derived measurements for flagged payloads at any subsequent time employing asynchronous scripts with no waste of computing resources.

6. Data sharing interfaces

There are different ways to provide users of the retePAIoT network with access to the transmitted data, both in real-time and at a later time. They consist in taking advantage of the mqtt functionalities, in creating push integrations and offering restful API interfaces.

To exploit the mqtt features, a new instance of mqtt server has been created and made available on a public ip address. The public mqtt server is tightly and bidirectionally linked with the private mqtt instance of ChirpStack by means of the mqtt bridge (Fig.4):

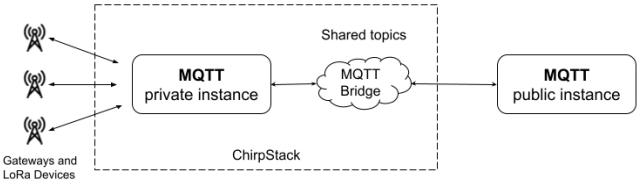

Figure 4: Public and Private instance of MQTT server

The mqtt bridge configuration requires to specify the topics that are to be shared; in this way, any message published on any of these topics on one of the mqtt instances is immediately sent to the other. The user, who has previously been provided with personal login credentials (username, password), has simply to connect to the public instance and subscribe to any desired topic. The login credentials are needed to profile the users in order to restrict the access only to their own data and keep all data private. Besides, the implementation design of dividing the devices into groups of 'applications', as defined on the ChirpStack application server, allows for simple and immediate management of data sharing with all authorized users. This solution highlights the advantage of using a public mqtt, that is to allow data sharing in a transparent and secure manner with all and only the authorized users.

It is worth noting that data are published only after having being decrypted, since the retePAIoT network is the owner of the devices and knows the communication keys for each device; that must be done immediately upon receiving a message because the keys are subject to change, ie the device can eventually renegotiate them at any time both in otaa and abp connection modes. Moreover, the private mqtt instance of the ChirpStack will receive all messages from any device that is in communication range of a gateway belonging to the retePAIoT network, so it is not unlikely

that it will receive a message from an unknown device. However, the public mqtt will receive all and only the messages managed by the retePAIoT network that must be shared. . The availability of a public MQTT directly connected to the private one allows to share even very large quantities of data in real time and in row form as if they were arriving directly from the private MQTT ChirpStack network server.

In our specific case of the retePAIoT network, an additional integration with an external InfluxDB database has been set up. For this integration to work, the user is required to notify the retePAIoT administrators for the settings to use. Once the server configuration is done - which in our case only necessitated editing a configuration file - the ChirpStack server automatically handles all the communication steps with the database, dramatically simplifying the process of acquiring and storing data on the database side.

InfluxDB is specialized in managing time series, which makes it a perfect match for IoT applications. When exporting the sensor data, the only attention should be paid to the building of the json data block with the prescription given by InfluxDB. In our specific case the aim of the external application is to monitor and analyse the information concerning the signal transmission quality, therefore only this part of data is being shared outside the retePAIoT and no specific adjustment was required to our json format. With this feature of the ChirpStack server the whole process of sharing the information with an external source was essentially effortless.

The retePAIoT project also offers access to data by means of the well-known restful APIs. Since APIs work on stored data, they can fetch measurements only after all of the previous modules have finished processing the incoming payloads. So far, the retePAIoT projects offers APIs to access the raw data (ie. the unprocessed message from the sensor) enriched with specific LoRaWan parameters like the RSSI, the SNR, the spreading factor and the used fPort as well as APIs to access the decoded value of any of the measurements of the sensor. The output is generated in the common json format for ease of use. For security reasons, moreover, when using the APIs the user must provide his personal access key in the body of the request. No data will be extracted without the auth key or if the requested sensor is not owned by the user identified by the auth key. The use of APIs for data retrieval is especially indicated for application purposes in the case of a limited number of sensors involved to exploit the availability of already decoded measurement values.

7. A significant data use case

Currently retePaIoT network is composed by seventy LoRa gateways, installed in fifty municipalities of Emilia-Romagna region; it manages almost one hundred type of sensors and their correspondents decoder through as many 'device profile' and ten 'application' to share data with mqtt interface to as many users; it integrates more than two-thousand sensors.

In this section a simple but significant use case of data from a single type of sensor integrated into retePAIoT is presented, which highlights the ultimate purpose of retePAIoT i.e. the use of sensors data which, in the specific case, concern the support to the rationalization of water resource.

Figure.5, in particular, shows a graphic relating to a water meter installed in a school immediately downstream of the multi-utility one, which accounts for daily water usage inside the building. Graphed data immediately highlight any losses, especially the large ones as in the case focused: the figure shows that during the first two week-ends, in the base line of the graph, when schools are closed, the daily consumption of water results more than seven cubic meters, evidently due to a loss in the piping system, and corresponding to almost 50% of the actual average daily consumption. The prompt detection of extra consumptions led to its subsequent resolution without further unnecessary waste, as shown in the graphic during following week-ends.

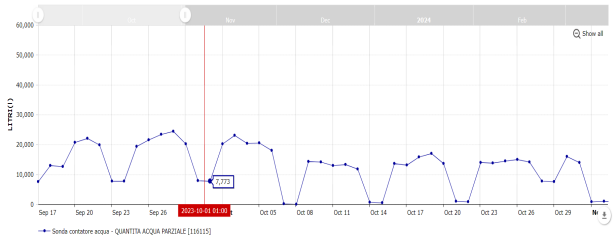

Figure 5: Graph of water consumption in a school

Figure.6 shows last, but not least, the installation of the water meter in the school, which by exploiting the LoRaWan interface and being battery-based, is particularly simple as well as the use of its data.

Figure 6: Installation of the meter water in the school

8. Conclusions

RetePAIoT is the public IoT network of Emilia-Romagna region, based on LoRaWan protocol, available for free to all public and private users interested in installing their own sensors of any type and for any purpose within the region.

Data generated by the sensors and collected by retePAIoT are made available through different interfaces, both real-time value and historical series, to the owners of the sensors, but, in aggregate and anonymized form, also to the Public Administrations for their own institutional purposes.

The original paper [1] had already highlighted the critical issues that the decoding process must manage in a public IoT network and the flexibility, robustness and scalability of the

solution implemented to solve them. Its evolution, presented in this paper, adds, however, a significant improvement of the decoding management of messages, because it based on a new native functionality of ChirpStack network server, on which retePAIoT is based, and on the javascript code normally provided by sensors manufacturers, without the need for new development for each type of new sensor and in favour of a greater efficiency. In particular, while the flexibility and robustness aspects of the solution presented derive mainly from having made the messages decoding phase asynchronous with respect to their reception, the evolution presented in this document has a positive impact especially with regards to the scalability of the service understood as the ability to manage new types of sensors efficiently in response to the increase of the market offers, maintaining unchanged performance, reliability and quality of service. The scalability is ensured, in fact, not only because, as in the original version, the management of the decoding modules grows linearly with the number of sensor types, and not with that of the sensors, but also because the possibility of using the java script code enormously reduces the time for the update of the service.

The improved model and architecture of data extraction and the different interfaces with which they are made available to all users and to third party platforms not only highlight the main goal of retePAIoT but they also provide an efficient and effective solution replicable for all IoT platforms which, like retePAIoT, have not only the objective of providing the collection, transport and storage of the messages of the sensors, but also that of the extraction of their data and the sharing for their use.

The data use case described at the end of the paper it's a demonstration of the importance of the use of data and therefore of the process of their extraction and sharing illustrated in this paper. It also shows how retePAIoT is an IoT infrastructure that can enable efficient and low-cost monitoring of various phenomena, processes and infrastructures, through which it is possible to detect and understand certain problems and act consequently to resolve them.

References

[1] S. Nanni, M. Carboni, G. Mazzini, "Flexible, Robust, Scalable Solution to Extract Information from IoT Public Network Sensors", Softcom 2023 – Conference

[2] Vangelista, Lorenzo et al. "Long-Range IoT Technologies: The Dawn of LoRa™." FABULOUS (2015)

[3] Web interface for managing an Internet of Things Public Network Elisa Benetti (LepidaScpA, Italy); Gian Paolo Jesi (Lepida ScpA, Italy); Gianluca Mazzini (LepidaSpA & UniFe, Italy), Sensornets 2019 – Technical Workshop.

[4] S. Nanni, M. Carboni, G. Mazzini, "PAIoT Network: a unique regional IoT network for very different applications ", Sensornets 2021 – Conference

[5] https://www.thethings network.org

[6] https://www.chirpstack.io (May 2022)

[7] LoRa specification provided by LoRa Alliance (2015). [Online]. Available: https://lora-alliance.org/about-lorawan, last retrieved 10 May 2019

[8] Benetti, E., Bonino, S., Odorizzi, A., Mazzini, G. (2014). Design of Data Centers for Public Administration. In SoftCom 2014 (pp. 1-5). IEEE.

[9] https://www.chirpstack.io/application-server/use/device-profiles/#custom-javascript-codec-functions

[10] P.P. Ray, A survey on Internet of Things architectures, Journal of King Saud University - Computer and Information Sciences, vol. 30, no. 3, pp 291-319, 2018, doi: 10.1016/j.jksuci.2016.10.003

[11] Jararweh, Y., Al-Ayyoub, M., Darabseh, SDIoT: a software defined based internet of things framework., J Ambient Intell Human Comput, vol. 6, pp 453–461, 2015, doi: 10.1007/s12652- 015-0290-y.