A Semantic-Based Middleware for Supporting Heterogeneity and Context-Awareness Within IoT Applications

Mohammed Lamnaour [1,*], Moundir Raiss [1], Yasser Mesmoudi [1], Yasser El Khamlichi [1], Abderrahim Tahiri [1], and Abdellah Touhafi[2]

[1] Sigl, Ensate, Abdelmalek Essaadi University, Morocco
[2] Department of Engineering Technology (INDI), Vrije Universiteit Brussel (VUB), Belgium
Email: mohammedlamnaour@gmail.com (M.L.); raiss.moundir99@gmail.com (M.R.); ymesmoudi@uae.ac.ma (Y.M.); yelkhamlichi@uae.ac.ma (Y.E.K.); t.abderrahim@uae.ac.ma (A.T.); abdellah.touhafi@ehb.be (A.T.)
*Corresponding author

***Abstract*—Internet of Things technology, or IoT, is changing people's lifestyles. Smartwatches, smart cars, smart homes, smart farms, and more – IoT has already been incorporated into a variety of products and services. To efficiently manage interactions between currently deployed smart things and applications, IoT vendors worldwide continually introduce different middleware platforms to meet application development requirements. Therefore, finding a suitable IoT middleware is a major issue faced by developers, especially when the system contains heterogeneous smart things and generates a vast amount of heterogeneous data. Most existing IoT middleware models do not satisfy all functional requirements and are tailored to specific system layers. To address these issues, this paper proposes a middleware model based on semantic web technologies and context-aware computing as an enhancement of the previously developed middleware MSOAH-IoT (A Middleware based on Service Oriented Architecture for Heterogeneity Issues within the Internet of Things). It uses a low-level ontology to automatically register, classify and then identify heterogenous smart things. The developed middleware provides a search engine to determine the appropriate smart object to respond to incoming requests of real-time measurements from the user/application layer.**

***Keywords*—internet of things, semantic web technologies, context-awareness, ontology, middleware**

I. Introduction

The integration of IoT solutions into our daily lives increases our capability to observe and report existing phenomena around us. Their primary goal is to enhance daily human interactions in various fields, including transportation, healthcare, industry, environment, logistics, security, home automation, farming, and more [1]. However, despite the high growth of IoT applications, this technology is not yet mature due to the absence of unified norms and standards. Therefore, to provide high-performance IoT solutions, research efforts should focus on addressing the most common challenges in IoT, such as security, energy consumption, data processing, high-level knowledge, and heterogeneity [2, 3].

- Security: The risk of home invasion is one of the most concerning aspects of IoT solutions for home automation. Since the system collects personal data such as the user's financial information, habits, and working hours, it becomes a prime target for hackers and malware. Many IoT devices do not support security updates and cybersecurity measures.
- Energy Consumption: Maintaining and ensuring connectivity to the network consumes a significant amount of energy. Most connected devices consume roughly the same amount of energy whether active or in standby mode, especially when they remain connected. Many procedures are being developed to ensure low energy consumption and optimize the device's battery life.
- Data Processing and Management: The rapid development of the Internet of Things and its integration into various fields, along with the proliferation of connected devices, has led to an increase in the amount of collected data. This increase makes the management, processing, and analysis of data increasingly challenging. There are several approaches to IoT data management systems. Some facilitate the integration of generated data, while others provide efficient storage and indexing of structured and unstructured data.
- High-Level Knowledge: Machines cannot understand information as intuitively as humans can. Hence, it is critical to discover knowledge from raw data by collecting, modeling, and reasoning within the context.
- Heterogeneity: Industries and research communities have introduced a large number of IoT applications, creating a vast amount of heterogeneous IoT data from various sources in different formats.

In our research work, we focus on issues related to data heterogeneity and high-level knowledge. We aim to

Manuscript received June 6, 2023; revised July 30, 2023; accepted September 4, 2023; published January 2, 2024.

address these challenges by developing a middleware capable of receiving data through the physical layer, managing and processing it within the middle layer, and making decisions in the high-level layer. To date, IoT middleware with sufficient stability and efficiency has not been realized yet [4]. One of the most commonly used solutions to overcome the heterogeneity issue is the implementation of semantics within IoT middleware. This approach provides common information models using heterogeneous sources of information that can interoperate using the same concepts and relationships [3]. The ontology and modeling capabilities provided by the Semantic Web enable resources in the Internet of Things to be described in a unified, machine-understandable manner. Semantic ontology can also effectively address the challenges posed by the dynamic expansion of physical network system resources [5]. Semantic interoperability refers to the ability of different parties to access and interpret unambiguous data, as connected objects can exchange data with each other and with other users [6].

In our work, we integrate Semantic Web and context-awareness technologies into the MSOAH-IoT middleware previously developed by our research team. This integration allows us to integrate smart services into the system. The developed middleware introduces a sensor-level ontology to automatically register and classify smart, heterogeneous objects based on their types, roles, and embedded sensors. Subsequently, we utilize context-awareness to extract knowledge from collected information data through the data processing module. Moreover, the request processing module permits responding to the user/application layer to determine the appropriate objects for real-time measurements.

The structure of this paper is as follows: Section I defines the basic concepts and provides an overview of related work. Section II presents the middleware architecture in detail. In Section III, we delve into the design and implementation of the proposed middleware. Section IV discusses the results through a case study. Finally, in Section V, we present concluding remarks and outline avenues for future work.

II. Basic Concepts and Related Work

A. Semantic and Context-aware IoT Middleware

Numerous achievements have been made in the design of IoT middleware; nevertheless, none of them comprehensively addresses all IoT challenges. Each approach has its unique strengths and weaknesses, highlighting the need for new and innovative approaches. To date, there have been excellent surveys in the field of IoT middleware, with references [6–16] providing an overview of the most recent semantic and context-aware middleware solutions for IoT, as summarized in Table I.

TABLE I. Summary of Semantic Middleware for IoT

Semantic middleware	Approach	Context-aware	Data management
OpenIOT	Cloud based middleware	yes	Data Storage
UbiROAD	Agent oriented middleware	yes	Data Storage
Ubiware	Agent oriented middleware	yes	Data Processing & Aggregation
Hermes	Event driven middleware	yes	Data Processing & Filtering
CHOReOS	Service oriented middleware	yes	Data Processing & Aggregation
Hydra	Service oriented middleware	yes	Data Storage
FIWARE	Service oriented middleware	yes	Data Storage
CA4IOT	Cloud based middleware	yes	Data Storage
CAMPUS	Service oriented middleware	yes	Data Storage
CASF	Service oriented middleware	yes	Data Storage
CoCaMAAL	Service oriented middleware	yes	Data Storage, Processing & Aggregation

TABLE II. Characteristics of MQTT, COAP, XMPP, AMQP and HTTP

Protocol	MQTT (Message Queuing Telemetry Transport)	COAP (Constrained Application Protocol)	XMPP (Extensible Messaging and Presence Protocol)	AMQP (Advanced Message Queuing Protocol)	HTTP (Hypertext Transfer Protocol)
Year	1999	2010	1999	2003	1997
Architecture	publish/subscribe	client/server	client/server	client/server	client/server
Header size	2 bytes	4 bytes	undefined	8 bytes	undefined
Transport Protocol	TCP	UDP	TCP	TCP	TCP
Security	SSL/TLS	DTLS	SSL/TLS	SSL/TLS	SSL/TLS
Quality of service/Reliability	QoS 0 - At most once QoS 1 - At least once QoS 2 - Exactly once	Confirmable, Non-confirmable	XML-Stanzas	Settle format, Unsettle format	Limited (via TCP protocol)
Encoding format	Binary	Binary	XML	Binary	Text
Licensing Model	Open source	Open source	Open source	Open source	Free
Organizational support	IBM, Facebook, Eurotech, Cisco, Red Hat,Amazon web service (AWS), InduSoft, Fiorano	Cisco, Contiki, Erika.	Facebook, Microsoft, Apple	Microsoft, JP Morgan.	Global web protocol standard.

OpenIOT is a cloud-based middleware for IoT that leverages the W3C Semantic Sensor Networks (SSN) ontology to enhance the sharing of common semantics among all IoT stakeholders. The key issues addressed by OpenIoT middleware include data management, service discovery, and semantic interoperability.

UbiROAD is specifically designed for smart traffic monitoring and employs a multi-agent architecture.

Ubiware is a multi-agent-based middleware that integrates software agents into each IoT resource. These software agents are responsible for controlling the state of the resource and employ ontologies to address interoperability issues.

Hermes is an event-driven middleware that employs a peer-to-peer architecture. It supports fault-tolerance, event discovery, event delivery, and security. However, it has limitations in terms of mobility and does not support composite events or persistent storage for events.

CHOREOS is designed to provide large-scale services for the Future Internet. It includes eXecutable service composition (XSC) for service composition, eXtensible service access (XSA) for IoT service access, and eXtensible service discovery (XSD) for discovering IoT/embedded services.

Hydra (LinkSmart): Initially funded by the European Union and later renamed LinkSmart in 2014, Hydra aims to integrate sensors into ambient intelligence systems. It treats IoT devices as services and utilizes a Service-Oriented Architecture (SOA) to manage IoT devices, events, context, storage, and security.

FIWARE is based on a public cloud platform and offers a rich library of modules known as Generic Enablers, which provide various added-value functions (referred to as services). Cognitive Enablers within FIWARE use semantically abstracted metadata through well-defined Restful APIs to make decisions.

CA4IOT (Context Awareness for Internet of Things) is a sensing-as-a-service middleware primarily focused on selecting the most suitable sensors based on specific tasks or problems, rather than providing a comprehensive middleware solution for managing context data.

CAMPUS (Context-Aware Middleware for Pervasive and Ubiquitous Service) is designed to automate context-aware decisions. It incorporates compositional adaptation, ontology, and description logic/first-order logic reasoning. **CASF** (Context-Aware Services Framework): CASF is built upon semantic web services, known for their support in automatic service discovery and integration.

CoCaMAAL short for Cloud-oriented Context-Aware Middleware for Ambient Assisted Living (AAL), serves multiple purposes, including context modeling for raw data, context data management and adaptation, context-aware service mapping, service distribution, and service discovery. *FIWARE, CA4IoT, CAMPUS, CASF,* and *CoCaMAAL* achieve the highest levels of context awareness. However, it's important to note that they may differ in their real capabilities, which are directly related to their respective levels of context awareness [14]. The desired awareness means that middleware could adequately understand any change of current environment. The majority of current context aware middleware proposals only reach a very limited level of cognition and awareness for their involved circumstances. Efforts should be put to reach higher levels of context awareness [11].

B. Communication Protocols

Several instant messaging (IM) protocols permit connecting devices in a distributed network and supporting the next generation of IoT applications. The most famous IM protocols used are MQTT and CoAP. These two protocols connect devices through small-sized messages and have lightweight message overhead [16]. Communication protocols such as HTTP, XMPP, and AMQP are implemented in various IoT applications. TABLE II demonstrates the advantages and disadvantages of these protocols, as compared and presented by authors in [17–19].

III. Architecture of the IoT System

There is an ongoing discussion about the layers in the architecture of the IoT. Some approaches consist of a three-layer architecture defined as the sensing layer, the network layer, and the application layer. Others include a four-layer architecture by introducing a service layer between the application and the network layer for data management.

According to the International Telecommunication Union, each IoT architecture should be divided into five layers: sensing, accessing, networking, middleware, and application. However, more advanced models, such as cloud-based architectures, and various six-layer models, can also be found in the literature [13].

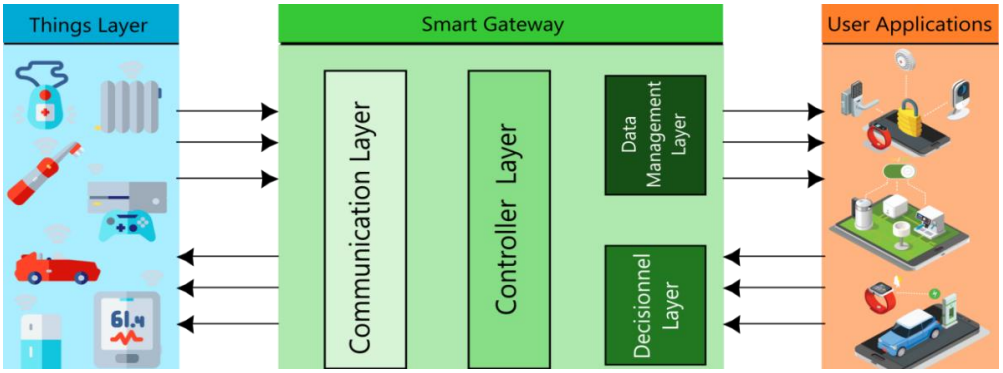

Fig. 1. Interactions between the proposed middleware layers and components.

The proposed architecture shown in Fig. 1 is designed to facilitate the collection of data from various sensors (e.g., wearable devices) and then manage this data to derive meaningful information. Additionally, it enables real-time interactions with the system and supports real-time decision-making by analyzing sensor values. Besides the Things Layer and the Application Layer, the proposed architecture consists of the following layers: Communication Layer, Controllers Layer, Storage Layer, and Decisional Layer.

Moreover, the system interacts with two external layers: The Users/Applications Layer and the Things Layer. It describes the roles, functionalities, and components of each layer, as well as their interactions, starting from the low-level Things Layer to the high-level end Users and Applications Layer.

A. Things Layer

The Things Layer is an external layer that represents physical and virtual objects, whether they contain sensors or actuators embedded within the environment. We define different types of messages exchanged between these objects and the smart gateway:

Registration-msg: This is an authentication message sent to the smart gateway during the initial pairing process. It allows objects to share their properties, such as ID, name, location, and communication interface.

Sending Data-msg: After registration, data collected from the environment is transmitted to the smart gateway via a Sending Data-msg.

Applying action-msg: The smart gateway sends this message to objects to trigger an action or request real-time measurements.

Unregistration-msg: When an object is powered off, it sends an Unregistration-msg to the smart gateway. This message enables the smart gateway to remove the registered object from the database."

B. User Layer/ Applications

This layer represents human users, applications or services interacting with the system.

C. Communication Layer

The Communication Layer permits the retrieval and sending of data from/to things. It should handle the heterogeneity of communication protocols (BLE, WiFi, etc.). The chosen protocol for transmitting data between the smart gateway and things is MQTT; thus, the principal component of this layer is the MQTT broker. The MQTT protocol runs over TCP/IP, allowing things equipped with WiFi to exchange data. For things equipped with BLE or ZigBee, an additional module must be installed within them to enable communication with the MQTT broker

D. Controller Layer

This middle-level layer is responsible for the following functionalities:

- Receiving data from the lower-level layer and transmitting information to the database or the end-user application.
- Processing data to identify information/measures from the data.
- Receiving requests from the decisional layer, allowing the identification of things when additional measures need to be collected for the decisional layer.
- Applying requests to send requests for measures or apply decisions.

E. Data Management Layer:

This layer manages information after the data formatting process through the controller layer. It serves as a database that can be used by both the decisional and application layers. It not only provides real-time values but also historical values for further processing in the decisional layer, utilizing machine learning algorithms or high-level ontologies. Diene *et al.* [20] has classified data into five categories as described in Fig. 2.

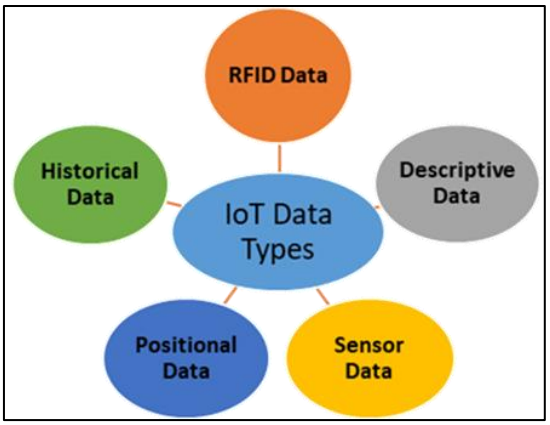

Fig. 2. Data types in IOT.

F. Decisional Layer

This component analyzes stored data, creates patterns, and uses predefined thresholds and rules to make decisions. For example, if an elderly individual is accustomed to opening the door at 7 o'clock and fails to do so, certain actions must be triggered, such as checking proximity sensors, sending alerts to his phone to determine his whereabouts, and monitoring his heartbeat, among others.

IV. Design and Implementation of the Proposed Middleware

A. Design of the Controller Layer

The controller layer in our architecture comprises two principal functions: data processing and searching for things. In this layer, we propose an ontology to describe the things installed within the environment. As depicted in Fig. 3, the controller layer is positioned as a middle-level layer between the data management layer, the decisional layer, and the communication layer.

We have designed a low-level ontology that describes things and their characteristics. The proposed ontology is updated through the data processing function, while the search things function extracts information and performs searches within the ontology based on SPARQL requests.

1) Module 1: Data processing

This module is responsible for retrieving received data from the communication layer, updating the designed ontology, and sending values to the Data Management Layer for storage.

Various types of messages are received and processed by this component. Each object/thing sends an identification message in JSON file format, which includes their ID, name, device type (fixed or ambient), topic (set to /device/on for this type of message), location where the device is placed, and user information if the device is associated with a person, such as in the case of a smartwatch. Finally, the message includes a list of sensors and actuators embedded in the device.

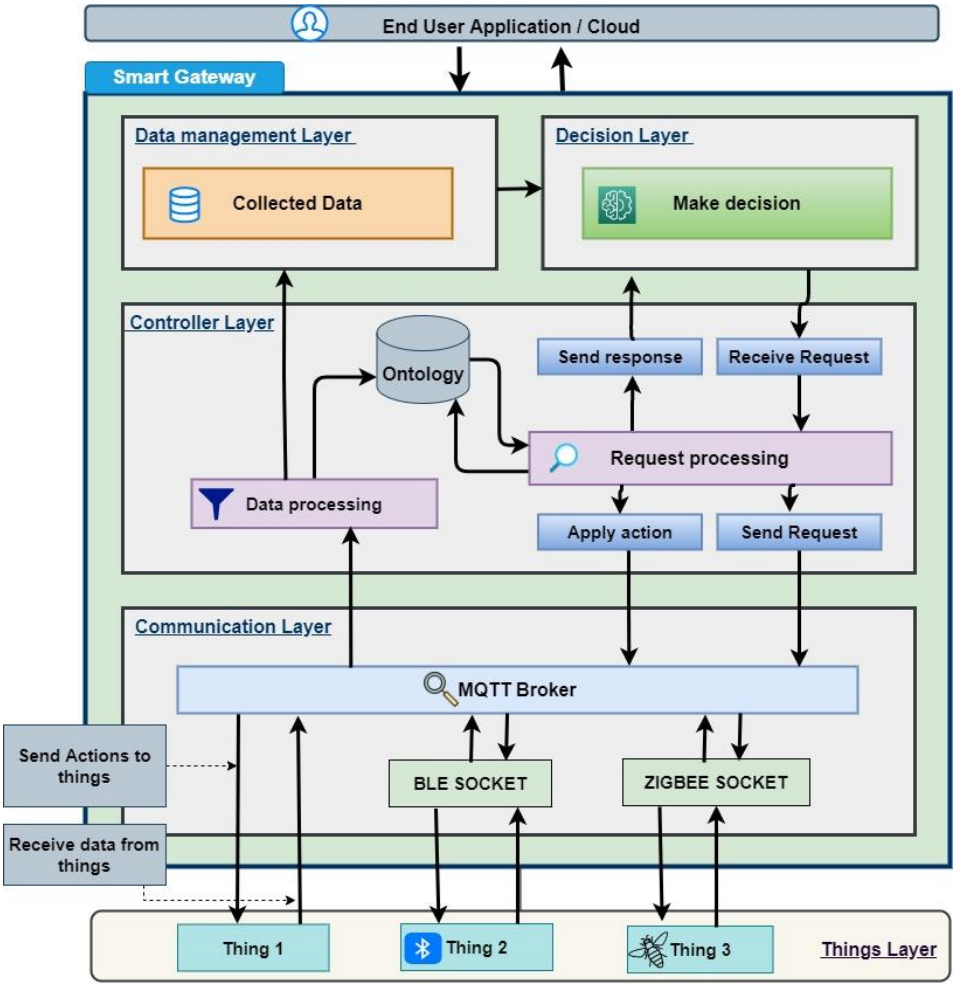

Fig. 3. Components of the controller layer.

For each detected event, the smart thing sends a message in JSON file format that contains the following information: topic, type (sensor/actuator), value of the measured phenomenon, date of detection, and the unit of measurement. This information is used to update the created ontology.

2) Module 2: Request processing

This module contains a search engine that allows it to determine the appropriate object and sensor to respond to incoming requests for real-time measurements from the decision layer. For instance, if the system requires the user's location or heart rate, the module provides a list of suitable things to fulfill the request by searching through the designed ontology.

B. Implementation Tools

Building IoT solutions in real life is not an easy task due to the heterogeneous nature of its IoT components. It is highly recommended, before developing an IoT application, to utilize simulation and testing tools at various stages [21]. This ensures the application's performance, reliability, and security [22]. Simulations are frequently employed to monitor the system's behavior over time, mitigating risks and avoiding the costs associated with real-world testing.

IoT simulation tools allow the evaluation of real-life scenarios in practice before implementing systems in operational environments. They are vital for various reasons, including ensuring the performance, efficiency, and reliability of applications. Numerous simulation tools are utilized in the IoT domain.

In our case, we use Node-RED, which is an open-source, flow-based development tool for integrating IoT hardware devices, Application Programming Interfaces (APIs), and online services. Node-RED is a free JavaScript-based tool built on the Node.js platform, providing a visual, browser-based flow editor [22, 23].

The programming language employed in our project is Python. Python is a high-level programming language known for its code readability and open-source licensing. Its simplicity allows developers to focus on problem-solving, requiring less code to achieve more. However, in our model, we need to access the ontology for consultation and modification. Python lacks a built-in library capable of such operations. Therefore, we use Owlready, a Python module designed for ontology-oriented programming. Owlready enables the loading of OWL 2.0 ontologies as Python objects, modification, and saving them to OWL XML format. It also facilitates reasoning through HermiT (included) [24].

C. Ontology Model

The proposed ontology serves two primary purposes: firstly, modeling the entities' environment, and secondly, establishing direct relations between the Application domain, Device, Location, and User. We argue that this relation is essential as it allows any IoT middleware to extract knowledge about the environment and classify every detected smart object. The process of designing the ontology consists of eight steps: Specify the domain – Consider reuse – List of terms – Specify classes – Specify properties – Define instances. The first step in ontology development is defining the ontology's domain and scope, answering some fundamental questions [25]: What subjects will be covered by the domain ontology? What is the use of the ontology? What types of questions could be answered by the information in the ontology? The answers to these questions may evolve during the ontology-design process, but they help limit the scope of the model [26]. The primary goal of our ontology is to represent and classify installed devices by application domain within the covered environment. It must be capable of addressing the following purposes: What is the list of covered devices? What are the characteristics of each device? Is the device linked to a location or user? What is the list of sensors, actuators, and protocols embedded in each device? What are the domains/sub-domains of the device application?

The second step involves considering the reuse of existing ontologies if the model needs to interact with specified ontologies or controlled vocabularies [26]. In our case, we have skipped this phase. The next step involves defining a list of terms. It is useful to list all the keywords or terms that we want to use, whether in the form of statements or explanations to the user [25]. What are the terms we would like to discuss? What properties do these terms have? What would we like to convey about those terms? These questions are based on competency and the elaboration of the questions in more detail. An example of a list of terms for the case of a smart home includes: Device/Sensor/Actuator/Domain/Sub-domain/User /Location/communication protocols.

Step four begins by defining classes. From the list created in Step three, terms are selected that either describe related objects or other objects with independent existence. These terms are classified in the ontology and become anchors in the class hierarchy. Classes are also organized into a hierarchical taxonomy. As shown in Fig. 4, the main classes of this ontology are:

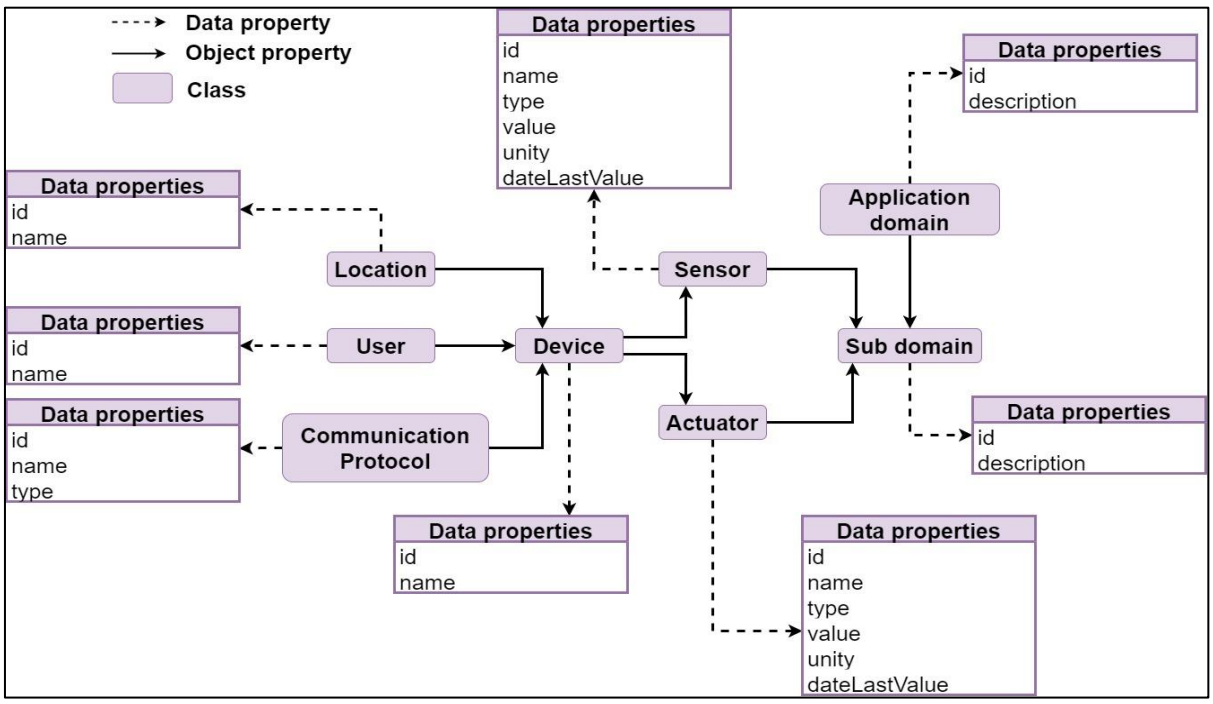

Fig. 4. Classes and properties of the proposed ontology.

- Domain: This includes all application domains covered by the received data. For example, the collected data in a smart home can provide information about Health, Security, Comfort, Energy, and Multimedia.
- Subdomain: Each domain has a list of subdomains, and each subdomain is associated with a unique application domain.
  1) Health: Body temperature, Vitals (Heart rate, blood pressure), accelerometer data, sleeping hours, and step count.
  2) Security: Intrusion detection, gas leak detection, and fire detection.
  3) Comfort: Light monitoring, outdoor weather information, air conditioner control.
  4) Energy: Electricity consumption, water consumption, gas consumption.
  5) Multimedia: Multi-room audio control.
- Device: This represents a real-world component that incorporates a list of sensors, actuators, and communication protocols.
- Actuator: An actuator allows a device to perform an operation or control a physical entity.
- Sensor: The sensor is the device's module that measures a physical property of the real world.
- Communication protocols: These protocols enable the device to communicate with other devices.
- Location: This class is used to determine the location of a device or the list of devices associated with each location. For example, the list of locations in a smart home may include the living room, guest room, parents' bedroom, children's bedroom, and kitchen.
- User: This class is used to determine the list of devices associated with each user.

Classes alone may not provide enough information to address the competency questions from Step one [26]. After defining some classes, it's necessary to clarify and reflect on the internal structure of concepts. The extracted properties are illustrated in Fig. 4, where "Individuals" represent views and display types of Class Assertions, which are instances of each class. In the proposed ontology, the system creates individuals automatically.

V. Results and Discussion

A. The Use Case

Using Node-RED, we simulated a smart home scenario, specifically focusing on kitchen monitoring. Fig. 5 illustrates how the list of devices in this use case was implemented in a flow within Node-RED. Once initiated, each device sends a single message to the MQTT broker (Mosquitto MQTT broker) with /device/on as the topic. This message includes a list of properties such as ID, name, a list of sensors, actuators, and embedded communication protocols.

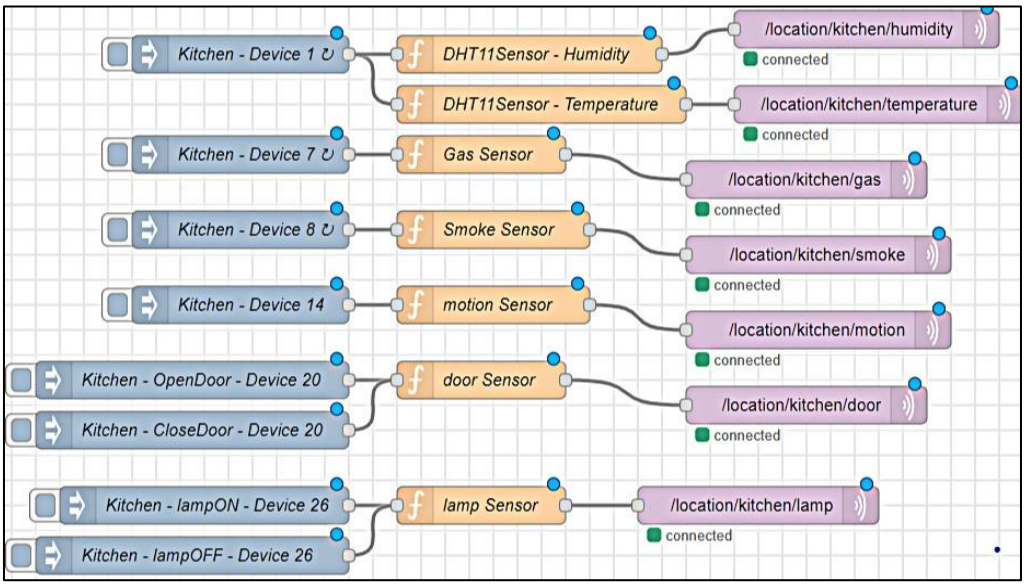

Fig. 5. Part of the Node-red Flow designed to represent the list of devices in the kitchen.

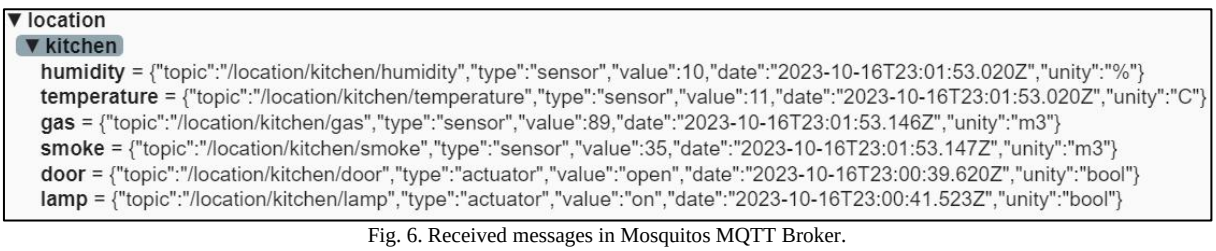

Fig. 6. Received messages in Mosquitos MQTT Broker.

In the proposed use case, we employed approximately 100 smart objects to simulate a smart home. We programmed many smart devices to send measurements of various phenomena, as demonstrated in Figure 5. These measurements included temperature (Celsius), humidity (percent), gas consumption (m3), smoke detection (m3),

and motion detection (yes/no). Additionally, we used actuators such as a door actuator and a lamp actuator (On/Off).

To monitor the messages generated by this Node-RED flow and received by the MQTT broker, there are various tools available, as presented in [27]. These tools can be highly useful when dealing with numerous devices and a multitude of topics. For tracking all the received messages and maintaining a history of these messages, we selected MQTT Explorer due to its ability to display message updates. Fig. 6 provides an example of messages captured by MQTT Explorer. Nodes in Node-RED are programmed using the JavaScript programming language. The developed program embedded within nodes enables the generation of random values between 0 and 100 for the kitchen's temperature. These values are then sent within the topic /location/kitchen/temperature to the MQTT Broker.

B. Results

Fig. 7 illustrates the ontology created automatically using Protégé once the system started. Fig. 8 displays a list of instances for the sensor class.

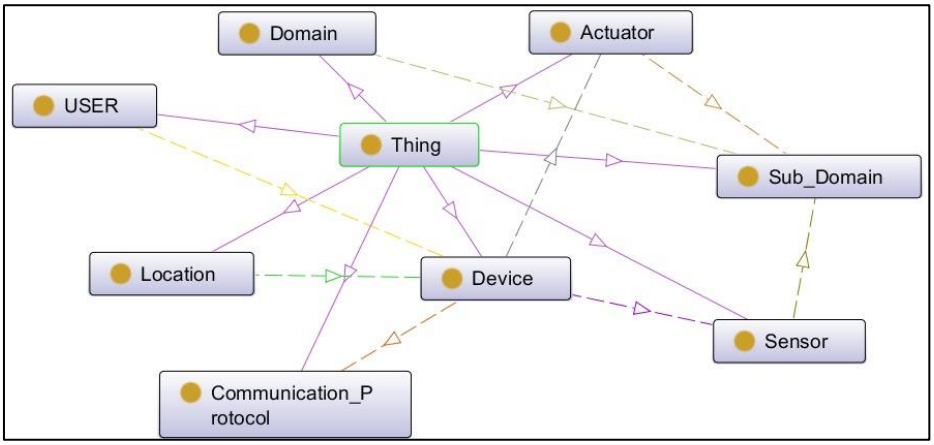

Fig. 7. The proposed Ontology generated by the system.

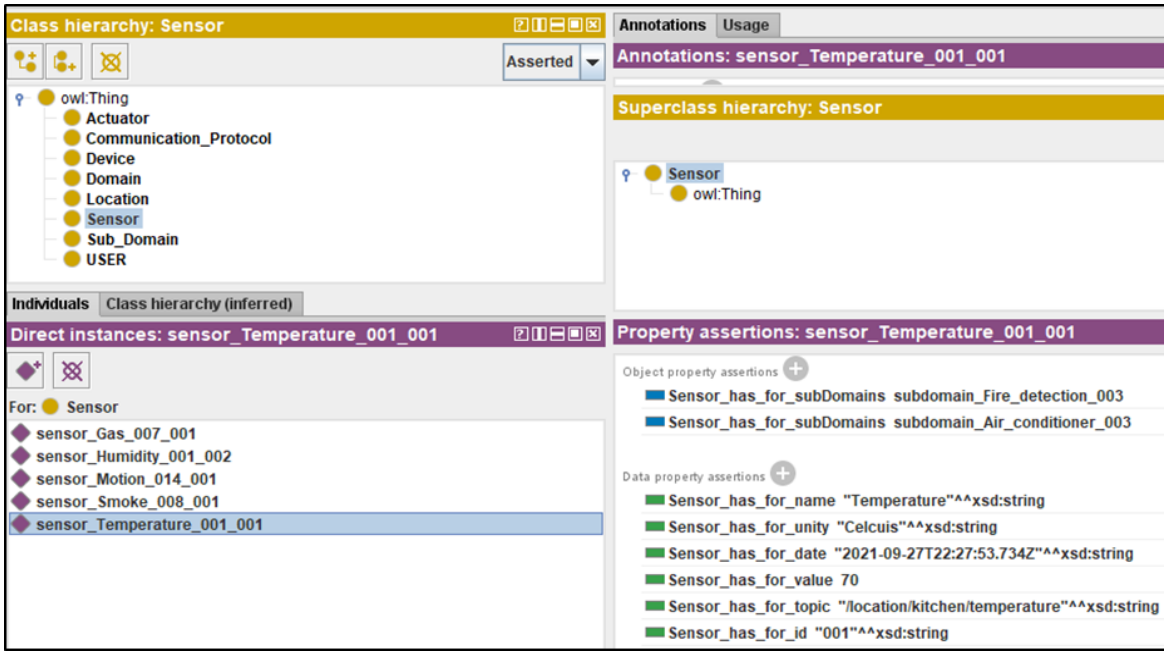

Figure 8. Properties of the temperature's sensor.

The system integrates all the information about the users in the environment, domains, sub-domains, devices, sensors, actuators, communication protocols, and locations within a JSON file. It then saves instances of all the sensors embedded within the smart objects in the same file format. In the last figure, the properties of the temperature sensor embedded within the instance 'device 1' are shown in detail. As demonstrated, the system correctly classified all devices and their properties as defined in the designed ontology. Data and characteristics were collected from smart things, and measures were extracted and transmitted to the database.

VI. Conclusion

The overarching goal of this work is to develop a smart gateway capable of automatically registering heterogonous smart IoT objects, collecting measurements, and processing data. In this paper, we introduced key concepts related to semantic web and context awareness technologies. We then conducted a survey of recent

middleware approaches proposed by the research community. Finally, we demonstrated the proposed middleware's architecture and implementation. The main components of the system are the data processing and the request processing, each developed around a low-level ontology model. The ontology implemented in the middleware enables the automatic classification of each registered object based on its type and role, ensuring high performance and rapid response within the system. The data processing module is responsible for retrieving data received from the communication layer, updating the designed ontology, and transferring measurements to the data management layer for storage. The request-processing module implement a search engine to facilitate the determination of appropriate objects and sensors to respond to real-time incoming measurement requests from the decision layer. In future work, we plan to focus on designing and implementing the decision layer within the proposed middleware. This layer will enable the smart gateway to detect anomalies based on user habits and predefined thresholds, make decisions, and act through IoT system actuators. We will utilize technologies such as Machine Learning algorithms and semantic web technologies to analyze and process stored and real-time data, allowing us to define user habits and make the entire system smarter, more reliable, and efficient.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Mohammed Lamnaour carried out the software programming. Moundir Raiss, Yasser Mesmoudi and Yasser EL Khamlichi helped in the testing of the code components. Abderrahim Tahiri and Abdellah Touhafi conceived the original idea. All authors contributed to the design of the research and to the writing of the manuscript. All authors had approved the final version.

References

[1] M. Javaid, A. Haleem, S. Rab, R. P. Singh, and R. Suman, "Sensors for daily life: A review," *Sensors International*, vol. 2, p. 100121, 2021.

[2] M. Lamnaour, M. A. Begdouri, Y. Mesmoudi, Y. E. Khamlichi, and A. Tahiri, "A semantic MSOAH-IoT design for improving efficiency and solving heterogeneity within IoT applications," *Journal of Communications*, vol. 17, no. 6, pp. 443-451, June 2022.

[3] T. Elsaleh, S. Enshaeifar, R. Rezvani, S. Acton, V. Janeiko, and M. Bermudez-Edo, "IoT-Stream: A lightweight ontology for internet of things data streams and its use with data analytics and event detection services," *Sensors*, vol. 20, no. 4, p. 953, 2020.

[4] J. Zhang, M. Ma, P. Wang, and X. D. Sun, "Middleware for the internet of things: A survey on requirements, enabling technologies, and solutions," *Journal of Systems Architecture*, vol. 117, 2021.

[5] G. Chen, T. Jiang, M. Wang, X. Tang, and W. Ji, "Modeling and reasoning of IoT architecture in semantic ontology dimension," *Computer Communications*, vol. 153, pp. 580–594, 2020.

[6] A. Rhayem, M. B. A. Mhiri, and D. J. F. Gargouri, "Semantic web technologies for the internet of things: Systematic literature review," *Internet of Things*, Vol 11, 2020.

[7] D. A. Cec, M. Novak, and D. Oreski, "Using semantic web for internet of things interoperability: A systematic review," *International Journal on Semantic Web and Information Systems*, vol. 14, no. 4, pp. 147–171, 2018.

[8] G. Fersi, "Middleware for internet of things: A study," in *Proc. IEEE Int. Conf. Distrib. Comput. Sens. Syst. DCOSS*, pp. 230–235, 2015.

[9] S. Hachem, T. Teixeira, and V. Issarny, "Ontologies for the internet of things," in *Proc. 8th Middleware Doctoral Symposium (MDS '11), Association for Computing Machinery*, pp. 1–6, 2011.

[10] N. Seydoux, K. Drira, and N. Hernandez, "Autonomy through knowledge: How IoT-O supports the management of a connected apartment," *Semantic Web Technologies for the Internet of Things*, 2016.

[11] E. M. Li *et al.*, "Context aware middleware architectures: Survey and challenges," *Sensors*, vol. 15, no. 8, pp. 20570–20607, 2015.

[12] P. Temdee and R. Prasad, "Introduction to context-aware computing," in *Proc. Context-Aware Communication and Computing: Applications for Smart Environment. Springer Series in Wireless Technology*, pp. 1–13, 2018.

[13] Q. Alfalouji, T. Schranz, A. Kumpel, M. Schraven, T. Storek, S. Gross, A. Monti, D. Muller, and G. Schweiger, "IoT middleware platforms for smart energy systems: An empirical expert survey," *Buildings*, vol. 12, no. 5, 2022.

[14] D. Rathod and G. Chowdhary, "Survey of middlewares for internet of things," in *Proc. 2018 International Conference on Recent Trends in Advance Computing (ICRTAC)*, pp. 129–135, 2018.

[15] R. Zgheib, E. Conchon, and R. Bastide, "Semantic middleware architectures for IoT healthcare applications," *Enhanced Living Environments. Lecture Notes in Computer Science*, vol 11369, 2019.

[16] T. M. Tukade, R. M. Banakar, "Data transfer protocols in IoT-an overview," *Int. J. Pure Appl. Math*., vol. 118, no. 16, pp. 121–138, 2018.

[17] B. H. C¸ orak, F. Y. Okay, M. G¨uzel, Murt, S. Ozdemir, "Comparative analysis of IoT communication protocols," in *Proc. Int. Symp. Networks, Comput. Commun. ISNCC 2018*, 2018.

[18] D. Bilal, A. U. Rehman, and R. Ali, "Internet of things (IoT) protocols: A brief exploration of MQTT and CoAP," *" Int. J. Comput. Appl*., vol. 179, no. 27, pp. 9–14, 2018.

[19] Y. Mesmoudi, M. Lamnaour, Y. E. Khamlichi, A. Tahiri, A. Touhafi, and A. Braeken, "A Middleware based on service-oriented architecture for heterogeneity issues within the internet of things (MSOAH-IoT)," *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 10, pp. 1108–1116, 2020\.

[20] B. Diene, J. J. P. C. Rodrigues, O. Diallo, E. H. M. Ndoye, and V. V. Korotaev, "Data management techniques for internet of things," *Mech. Syst. Signal Process*., vol. 138, 2020.

[21] E. Ojie and E. Pereira, "Simulation tools in internet of things: A review," in *Proc. 1st International Conference on Internet of Things and Machine Learning IML '17*, pp 1–7, October 2017.

[22] M. Ashouri, F. Lorig, P. Davidsson, R. Spalazzese, "Edge computing simulators for iot system design: An analysis of qualities and metrics," *Futur. Internet*, vol. 11, no. 11, 2019.

[23] M. Lekic, G. Gardasevic, "IoT sensor integration to Node-RED platform," in *Proc. 17th International Symposium INFOTEH-Jahorina (Infoteh)*, pp. 1–5, 2018.

[24] J. Lamy, "Owlready: Ontology-oriented programming in Python with automatic classification and high-level constructs for biomedical ontologies," *Artificial Intelligence in Medicine*, vol. 80, pp 11–28, 2017.

[25] S. Nandhinidevi, K. Saraswathi, M. Thangamani, and M. Ganthimathi, "Design and development of bird ontology using protégé," *Mater. Today*, pp. 1–6, Mar 2021.

[26] N. F. Noy and D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology," *Stanford Knowl. Syst. Lab.*, p. 25, 2001.

[27] A. C. Cristian, T. Gabriel, M. Arhip-Calin, and A. Zamfirescu, "Smart home automation with MQTT," in *Proc. 54th International Universities Power Engineering Conference (UPEC)*, pp. 1–5, 2019.