


RESEARCH

Open Access



Towards data sharing economy on Internet of Things: a semantic for telemetry data

Dareen K. Halim^{*†}  and Samuel Hutagalung[†]

*Correspondence:

dareen.halim@umn.ac.id

[†]Dareen K Halim and Samuel

Hutagalung are equally
contributed to this work.

Dept. of Computer
Engineering, Universitas
Multimedia Nusantara,
Jl. Scientia Boulevard,
Tangerang, Indonesia

Abstract

Internet of Things (IoT) provides data processing and machine learning techniques with access to physical world data through sensors, namely telemetry data. Acquiring sensor data through IoT faces challenges such as connectivity and proper measurement requiring domain-specific knowledge, that results in data quality problems. Data sharing is one solution to this. In this work, we propose IoT Telemetry Data Hub (IoT TeleHub), a general framework and semantic for telemetry data collection and sharing. The framework is principled on abstraction, layering of elements, and extensibility and openness. We showed that while the framework is defined specifically for telemetry data, it is general enough to be mapped to existing IoT platforms with various use cases. Our framework also considers the machine-readable and machine-understandable notion in regard to resource-constrained IoT devices. We also present IoThingsHub, an IoT platform for real-time data sharing based on the proposed framework. The platform demonstrated that the framework could be implemented with existing technologies such as HTTP, MQTT, SQL, NoSQL.

Keywords: Internet of Things, Data sharing, Data semantic, Interoperability, Platform, Framework

Introduction

Data plays a significant role in allowing businesses to gain competitive advantages and enabling researchers to develop new and better insights, algorithms, and technologies [1–3]. With the advancement of Internet of Things (IoT) technologies and its growth in adoptions [4, 5], the number of physical world data measured and digitized is also increasing [6].

Telemetry data is one particular type of physical world data widely used and associated with sensor readings in IoT applications [7]. Numerous works are using telemetry data with machine learning or other analytic methods for various domain-specific applications, such as predictions of energy demand on smart grid systems [8–11], room ambient modeling for green buildings [12–14], and indoor positioning [15–18].

The process of logging (or collecting) physical world data with IoT faces many challenges, one of them being that proper physical measurements mostly require domain-specific knowledge. For example, measuring room temperature and humidity for efficient HVAC conditioning purposes requires proper sensors' positioning and

calibration [19, 20]. Similarly, in the agricultural field, the measurements of the condition of the crops must also be appropriately performed lest the collected data may not provide any helpful insight [21–23]. Another example is energy consumption monitoring, where some methods require tapping into the power line, which may introduce hazards if done inappropriately [24, 25].

Connectivity is also one of the main problems with IoT data logging, where we expect IoT devices to maintain periodic logging when facing disconnections [26–28]. ‘Store-then-send when the connection is available’ is a standard method but is not always applicable to IoT devices with limited resources. Failure to implement proper countermeasures against disconnection may result in missing sensor data points, resulting in difficulties during data processing and analysis [29–31]. While methods like data imputation [32–34] can alleviate this problem, complete data points are always more beneficial. Paired with the lack of domain-specific knowledge during measurements, it may compromise the data quality even further.

Data sharing is one possible solution to the data quality problem. However, extensive works on IoT systems and IoT architectures led to a vast range of devices, architectures, logical elements, and data formats. Data sharing itself hence boils down to a semantic interoperability problem [35]. W3C Semantic Sensor Network (SSN) [36] is one widely used ontologies conforming to W3C Semantic Web technologies for representing elements in sensor networks. FIESTA-IoT proposed an IoT ontology by combining various aspects from SSN, DUL, IoT-lite, WGS84, Time, and M3-lite taxonomy [37]. Gyrard et al. [38] devised a formal methodology namely SEG 3.0 for unifying IoT data, enriching those data with semantics, and representing them as services. UbiRoad [39] worked on application-specific semantics for intelligent transportation systems by defining semantic interoperability between car-embedded devices and roadside devices. Semantics for time series data is proposed in [40]. They developed DS-Ontology, a semantic model for IoT data streams, along with the required tools added to existing Time Series Databases (TSDBs). Matos et al. [41] views data sharing and interoperability as a subset of context sharing, in that context sharing emphasizes the understandability of data for human users. Their review of a large number of IoT context-sharing platforms shows that the widely varying platform architecture can be broken down into general building blocks.

From a broader view, the works mentioned above on semantics also promote virtualizing devices and separating hardware from software. This idea, introduced as IoT virtualization, is a concept where different services and applications share the same underlying infrastructure, particularly IoT devices [42]. It allows faster development of applications, as well as reducing the cost associated with deploying hardware. Works such as [42–44] propose various architectures for virtualizing IoT devices as services. By sharing devices, IoT virtualization inherently also promotes data sharing. Scalability, efficient resource utilization [42, 43] and reliability [45] are other key factors promoted by IoT virtualization.

We observe two things from the previous works on data semantics and IoT virtualization. First, works that propose general semantics and architectures tend to be very broad thus do not define particular data structures aside from the semantics. Meanwhile, application or data type-specific semantics tends to lose generality in their discussion.

Second, while they propose general semantics and architectures, most define their generality in terms of application use case.

In this work, we narrow our focus to data sharing framework and semantic specifically for telemetry data. We kept our framework and semantic, IoT Telemetry Data Hub (IoT TeleHub) as general as possible. Thus, the contributions of this paper are as follows:

- We proposed IoT Telemetry Data Hub (IoT TeleHub), a general framework and semantic for collecting and sharing telemetry data. This is discussed further in later sections in which we build our framework and semantic on three principles, (a) abstraction, (b) layering of elements, and (c) extensibility and openness.
- We demonstrate the universality of the proposed framework and semantic by mapping it to several IoT data sharing and virtualization platforms.
- We present a prototype that implements the proposed framework and semantics. The prototype serves as a proof-of-concept for the framework.

The rest of this paper is structured as follows. Section "[Related works on IoT virtualization and sharing](#)" discusses previous works and their implications for this work. We present our proposed framework and semantic in "[Proposed framework and semantic—IoT TeleHub](#)" section and discuss the mapping into existing data sharing and virtualization frameworks in "[Mapping of existing virtualization and sharing platforms](#)" section. The prototype implementation is discussed in "[Prototype as an implementation of the framework](#)" section. We describe our works view on machine-readable and machine-understandable notion in "[Discussion](#)" section. Lastly, "[Conclusion, limitation and opportunities for future research](#)" section concludes this paper.

Related works on IoT virtualization and sharing

This section reviews related works, where we first discuss about interoperability of IoT platforms, then followed by related works on IoT virtualization and data sharing. We discuss the implication of the earlier works for our work, i.e., how we incorporate the main ideas of those works into our proposed framework and semantic.

Interoperability of IoT platforms

There is a large and growing number of IoT platforms proposed by industries and academics. Alongside, there are also growing numbers of infrastructure, architecture, protocols, standards, semantics [35] that are local to each platform or some of the platforms. While this variability allows better innovations to flourish, it also introduces IoT interoperability problems. [35] classifies IoT interoperability into (1) device, (2) network, (3) syntactical, (4) semantic, and (5) (cross) platform interoperability problems. Among these problems, we see that solving semantic and cross-platform interoperability provides a faster approach to unifying the varying platforms. Both semantic and cross-platform interoperability can be defined at higher abstraction levels. Hence changes to existing platforms can be kept minimal.

W3C defines Semantic Sensor Network (SSN) and is one widely adopted ontology, aiming at interoperability between IoT platforms that implement the standard [36]. FIESTA-IoT combines aspects from various ontologies from W3C's SSN, IoT-lite, DUL,

WGS84, Time, and M3-lite taxonomy [37]. It emphasizes on annotating the different ontologies used by the IoT platforms to reduce required changes and maximize interoperability. Gyrard et al. [38] proposed SEG 3.0, a complete and formal methodology for unifying IoT data with semantic web technologies, applying context information on the data, and finally representing them as services. This complete methodology ensures semantic interoperability but requires changes on multiple layers and elements for existing IoT platforms.

De Matos et al. [41] highlights the importance of context information for data, and proposes a general architecture for context sharing platforms. The architecture's complexity is due to its attempt to cover different data formats, context formats and applications. Zeng et al. [40] narrow their focus on semantic models for time series data, considering time series databases (TSDB) as the underlying storage solution. They developed SE-TSDB, a semantic-establishing tool suite that runs on top of TSDBs. While this solution works well on platforms with TSDBs as their storage solution, it may lose generality when we consider a more extensive range of platforms that use other storage technologies.

Based on the previous works above, we infer these crucial points to incorporate into our framework and semantic:

- Semantic and cross-platform interoperability at higher levels of abstraction.
- Minimize changes to existing platforms and semantics.
- Trade-offs between case-specific semantics and generality.

IoT virtualization and data sharing

In this subsection we discuss various works on IoT architectures that promote data sharing and virtualization of IoT elements. Benazzouz et al. [44] introduces a general system architecture for sharing IoT devices. It consists of (1) a heterogeneous layer where IoT devices reside, (2) a middleware layer that covers an IoT gateway functionality for handling the heterogeneity of IoT devices, and (3) an application layer that provides uniform access for end-users. The architecture regards IoT devices as *services* from which end users can access the device resources.

FogFlow [43] is a framework for scalable programming of IoT services that utilize both cloud and edge computing. It introduces the concept of a *virtual device* to integrate existing yet varying IoT devices into the framework. In short, the existing devices can be ported to the FogFlow framework by representing them as *virtual devices* according to FogFlow format.

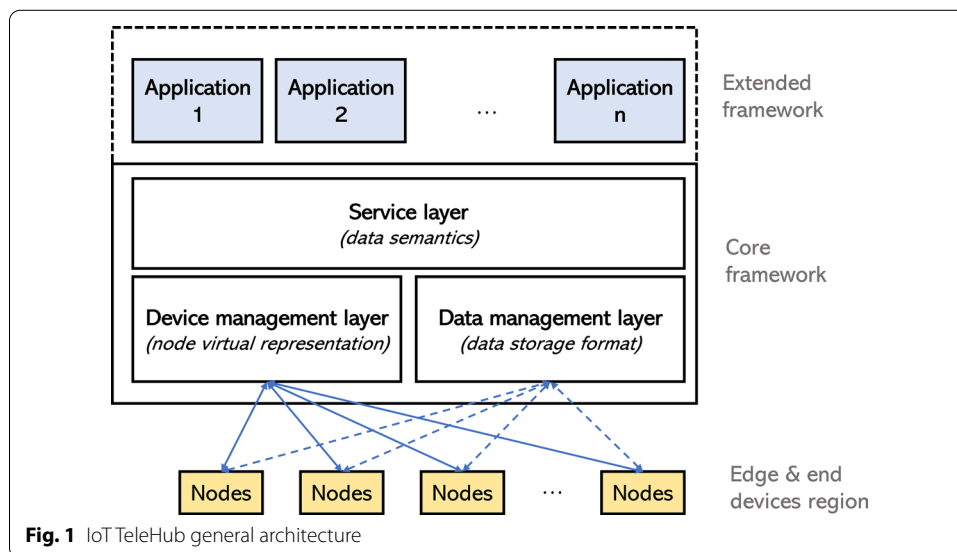
Gyrard et al. [38] through their SEG 3.0 methodology described a set of formal steps to build interoperable systems that work with heterogeneous data sources. The first two steps in that methodology are the *composing* step which transforms data with heterogeneous formats into a common format, and the *modeling* step that annotates the data with semantic web technologies. They treat IoT devices virtually as data producers that provide certain data formats that must be taken into consideration in the *composing* step.

Samaniego et al. [46] represents various elements in an IoT system as virtual resources, which are abstractions of physical and virtual entities. Based on that concept of virtual resources, they propose an IoT architecture consisting of three layers, i.e., the physical layer, the hardware abstraction layer (HAL), and the view abstraction layer (VAL). The physical layer is the actual IoT devices, represented virtually in the system as HAL. By representing the actual IoT devices as HAL, the system gains a uniform access to all the devices, easing the management of those devices. The end users of the system see the accessible resources through the VAL that controls what users can see and with which access levels. The work highlights the concept of abstracting system elements to simplify the management of resources.

Ogawa et al. [42] proposes a similar concept of IoT virtualization for efficient resource usage in the context of a smart city. They propose an architecture for applications to share IoT nodes by consolidating the nodes data and exposing them as *functions*. Specifically, these three layers comprise the architecture; (1) Gateway, on which the IoT nodes data are pooled and possibly pre-processed, (2) Function layer that provides access to the pooled data in certain formats, and (3) Application that utilizes the Function layer. The architecture also incorporates containerization to improve resource utilization and performance.

We derive the following points from the above-mentioned works.

- *Virtual device* concept to address IoT devices heterogeneity, and due to the ease of management of virtual representation of resources.
- Clear layering of services and their responsibilities.



Proposed framework and semantic—IoT TeleHub

In this section, we start discussing our proposed framework, IoT TeleHub, by its general architecture. We then discuss how the framework promotes the key aspects, (a) abstraction, (b) layering of elements, and (c) extensibility and openness in the said architecture. Figure 1 depicts the IoT TeleHub proposed framework. Nodes are physical IoT devices connected to the core framework. To maintain generality, we do not specify the connection method between the nodes and the core framework or whether the core framework is implemented on-site or in the cloud. Intermediary elements such as gateways are grouped in the “edge and end devices region”.

The core framework consists of three main elements as follows.

- Device management layer: Deals with the virtualization of the nodes. This layer is responsible for the hardware-software abstraction of IoT devices by defining the *node virtual representation*.
- Data management layer: Responsible for the “data channel” to which the nodes send their telemetry data. This layer handles nodes initial connection, nodes authentication, and the telemetry data sent by the nodes. To perform these tasks, it relies on the *node virtual representation* defined in the device management layer. This layer defines the *data storage format* for the telemetry data.
- Service layer: Provides service to applications, which at its most basic form is access to the telemetry data in a usable and shareable format. This layer defines the *data semantic*, separated from the data management layer’s *data storage format*.

IoT applications sit on top of the core framework, either separated from the framework or as a part of the framework, i.e., the extended framework. The extended framework part is optional. We include the applications as the extended framework to preserve the generality of the whole framework. That is, we intend to have this framework applicable to a wide range of IoT platforms. Some IoT sharing platforms may only provide elements up to the service layer, allowing users to build on top of the provided services. In that case, the extended framework element is unnecessary. Other platforms may include their applications on top of their services, regarded as the extended framework. Hence, by separating the application-specific aspect into the extended framework, we can define the layers in the core framework to cover only main functionalities shared by all IoT platforms.

From here onward, we use the term *framework* to refer to our proposed framework, IoT TeleHub, and the term *platform* to refer to existing IoT platform.

Virtual node representation

IoT devices are represented logically as virtual nodes in our *framework*, allowing the separation of the actual hardware details from the system. This approach provides two benefits, (1) abstraction and (2) security. As the devices are represented as logical entities, different hardware can be dealt with in the same manner, i.e., we can represent an ARM-based microcontroller and an Intel-based mini PC similarly. Masking hardware details fortifies the devices against the reconnaissance step of security attacks. Masking hardware is by no means a complete security solution as we cannot solely rely on

the obscurity of hardware details, and other security measures have to be incorporated. Discussion on more robust security measures is out of the scope of this work. A more powerful IoT device may take advantage of logical representation by representing itself as multiple nodes. For instance, a Raspberry Pi 4 is connected with several sensors that are grouped logically. We can run multiple processes on the Raspberry Pi; each accesses a different group of sensors and connects with the system as a different virtual node. Again, this provides complete abstraction of the underlying hardware while allowing logical separation of sensor arrays to ease the analysis later.

We describe our virtual node representation format as shown in Fig. 2. There are three mandatory fields defined in the format.

- *Uniquely namespaced ID*. Required to correctly identify every node in the *framework*, particularly for data sharing.
- *Access token*. Provides virtual nodes with access to the *data management layer*.
- *List of measurements*. Measurements correspond to the physical phenomenon measured by the node, e.g., temperature, humidity, occupancy, wind velocity, current consumption, and other telemetry data. Each measurement is defined by *label* and *tag* fields. The *label* field contains a string used to identify the measurement in a human-understandable format. The *tag* field contains a string that distinguishes the telemetry value sent by the nodes. This scheme helps in describing the data in a human-understandable format while allowing the most efficient data format for transmission. For instance, as illustrated in Fig. 2 we can use the *label* “Average Temperature” to describe our ambient temperature measurement, where the IoT node annotates the value it transmits with the *tag* “t”. Saving several bytes of characters in resource-constrained IoT setups is significant. For example, in a SigFox enabled device where the maximum uplink payload size is 12 bytes [47].

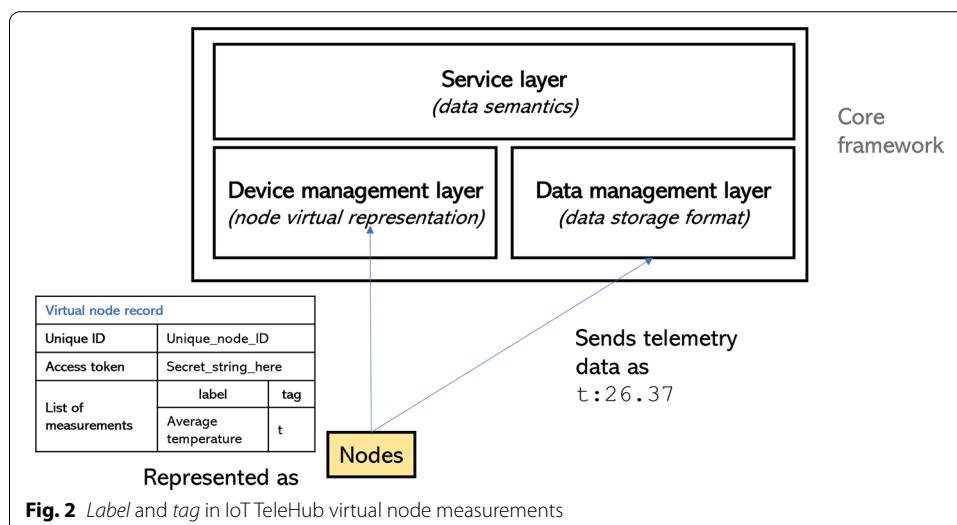
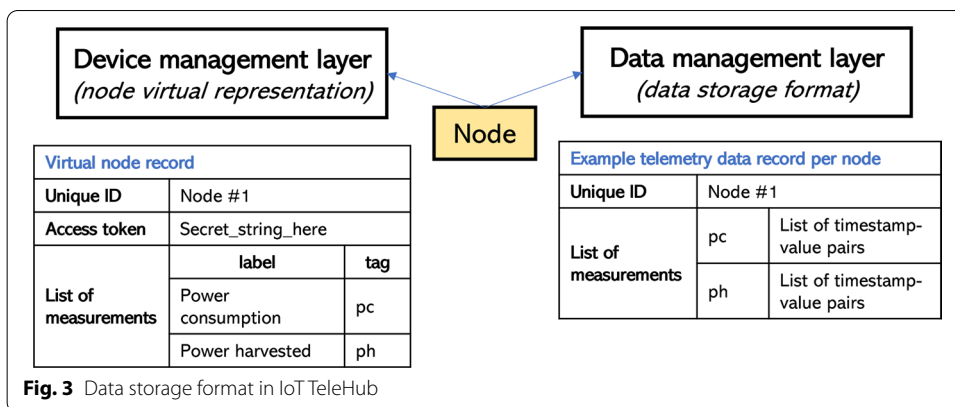


Fig. 2 Label and tag in IoT TeleHub virtual node measurements



Data storage format and layering of concerns

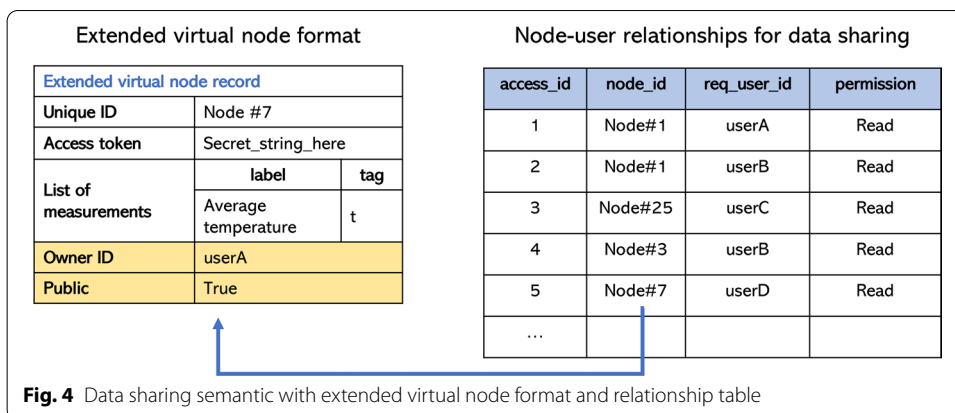
In the IoT TeleHub framework, we characterize telemetry data as time-series data, i.e., one-dimensional data. Data for each measurement on each node is stored separately, allowing flexibility in the logging method. That is, we store each measurement’s telemetry data as an array of *ts-val* pairs. *ts* is the timestamp and *val* is the value of the telemetry data. With this structure, an IoT node may have different sensors logging at different periods. Figure 3 shows an example of the data storage format for an energy harvesting controller. The node maintains two measurements, the “Power Consumption” and the “Power Harvested”, represented in a virtual node format shown in the left side of the figure. Data for each node is stored in a distinct object keyed by the unique identifier (should be identical to the *virtual node representation*). Furthermore, data for each measurement is stored in an array keyed by the measurement’s *tag* field. During operation, the node may send the two measurements together or separately as seen fit.

Service layer and data sharing semantic

Our *framework* broadly defines the service layer as the data translation layer but does not define translation formats as it limits generality. For instance, consider the case where the service layer needs to provide the stored measurements to an application layer. It could directly return the list of measurements keyed by the *tag* as shown in the data storage format in Fig. 3, and let the application layer query further which *label* does the *tag* correspond to from the virtual node record. Another way is the service layer could perform the *label* and *tag* matching from the virtual node record, then send it along with the measurement data to the application layer. Despite the trivial difference, this example illustrates how different platforms may adopt different data translation formats. Thus, defining an exact data translation format will remove the generality of our service layer.

We propose data sharing functionality on the service layer by defining *data sharing semantic*. We describe data sharing functionality as allowing *read* access to nodes data owned by specific users. In other words, the sharing is on a node basis. For that, we

1. extend the virtual node representation with *owner identifier* and *public flag* fields,
2. introduce sharing relationships between nodes and users, and



3. define a set of methods for data sharing.

Owner identifier describes the owner of a node, while *public flag* specifies whether a node is open for sharing. Figure 4 depicts this idea. Each entry in the node-user relational table is described as follows.

- *access_id*: a unique identifier for each node-user relationship.
- *node_id*: the unique identifier of the node being shared.
- *req_user_id*: the unique identifier of the user being granted access to the node data.
- *permission*: read or write permission. Earlier, we described that data sharing functionality is limited to read. This field is left for future use if write or more complex permissions are required.

We define our *data sharing semantic* to allow explicit sharing. That is, users can declare the discoverability of their nodes as public or private. Private nodes are only visible to their owners, while public nodes can be discovered by other users and have their data shared. We define the following methods to complete the *data sharing semantic*:

- *Set_node_public/private*: set nodes discoverability as public or private.
- *List_public_nodes*: list all nodes whose discoverability is public, along with selected fields of the virtual node representation.
- *Request_access*: request data sharing access for a specific node. The request will be shown to the user who owns the node.
- *List_access_requests*: list all data sharing access requests for a user.
- *Grant_access*: grant data-sharing access for a particular node to the requesting user. Create a new entry in the relationship table.
- *Reject_access*: reject data sharing access for a particular node.

With this data sharing semantic, the data sharing process is real-time and flexible. It removes the need to grant access to the physical devices to the requesting parties or

manually download the data and send them over. Please note that the data sharing semantic does not interfere with the translation functionality of the service layer. IoT platforms may adopt this data sharing framework and define their service layer’s data translation on the same level as the data sharing. We explore this further in our prototype described in the ‘[Prototype as an implementation of the framework](#)’ section.

Extensibility and openness

Summarizing the proposed framework and semantic, we have defined:

1. Device management, data management, and service layers in the general architecture. We have discussed each layer’s basic functionalities and the respective data formats.
2. Extensible data formats. We consider most fields in the data formats as metadata, which allows the incorporation of more functionalities to the framework by adding more metadata to the basic formats presented in this work. For instance, we can extend the *virtual node format* by adding node description, sensor types associated with each measurement, unit metrics, or location identifiers to better index IoT nodes [48]. Similarly, we can add measurement metadata to the *data sharing semantic* to have more fine-grained access sharing.

The proposed framework and semantic promote openness for interoperability between various platforms and applications, shown in Fig. 5. First, the service layer can allow applications to access required data through public application programming interfaces (API). Second, the data sharing functionality we defined in the service layer can also communicate with the data sharing functionality in other platforms. We can attain cross-platform sharing by exchanging the output of the *List_public_nodes* method, appended with a unique identifier of each platform.

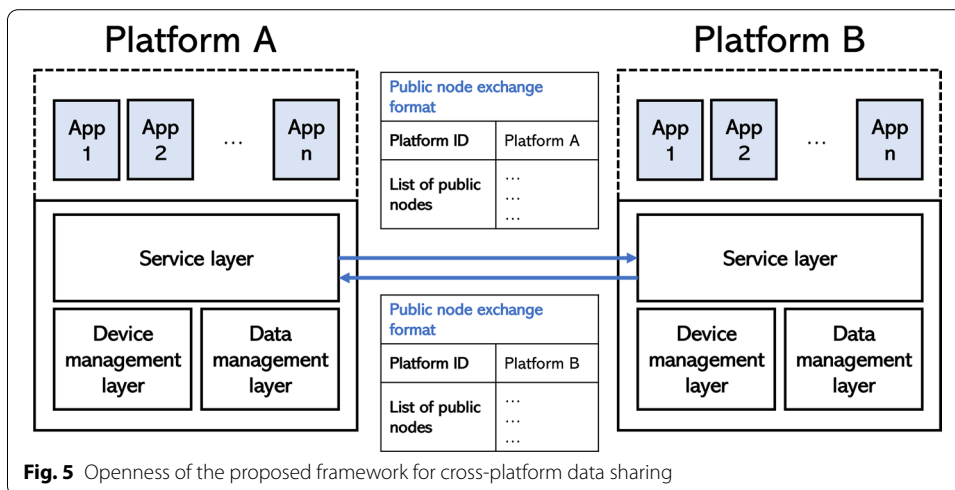


Fig. 5 Openness of the proposed framework for cross-platform data sharing

Different data schema is a very possible case for cross-platform data sharing, hence we describe further how our data sharing semantics approaches such a problem in the following two scenarios.

- *Data sharing in the same platform/framework (ours).* The proposed framework treats telemetry data as collection of one-dimensional time series data, i.e. arrays of timestamp-value pairs. In that sense, the general data schema is rigid. However, we allow flexibility in terms of users can freely specify what and how many ‘data fields’ to be stored in the system, per node. The data sharing semantic only defines whether other users can access the data from other user’s nodes. There is no difference in the stored data format aside from the aforementioned ‘data fields’. This applies to other platforms that fully adopt our framework as well.
- *Cross-platform data sharing.* We assume an extreme case of platforms storing telemetry data with a schema different from our framework. In that case, those platforms can still adopt our data sharing semantics since it only specifies whether certain nodes are available for sharing and which users have access to the shared nodes, decoupled from how the data schema is represented to other platforms. The *service layer* is the one responsible for translating the different data schema between platforms, commonly in the form of API calls.

Mapping of existing virtualization and sharing platforms

In this section, we map our *framework* to three architectures or frameworks introduced in previous works by Gyrard et al. [38], Ogawa et al. [42], and Guth et al. [49]. These three works were selected as our framework’s mapping targets due to their architectures covering aspects that we intend to highlight from our framework, which is detailed further throughout this section.

The mapping is only intended to demonstrate the generality of our *framework* and does not imply that our *framework* is better than those proposed in other works. In

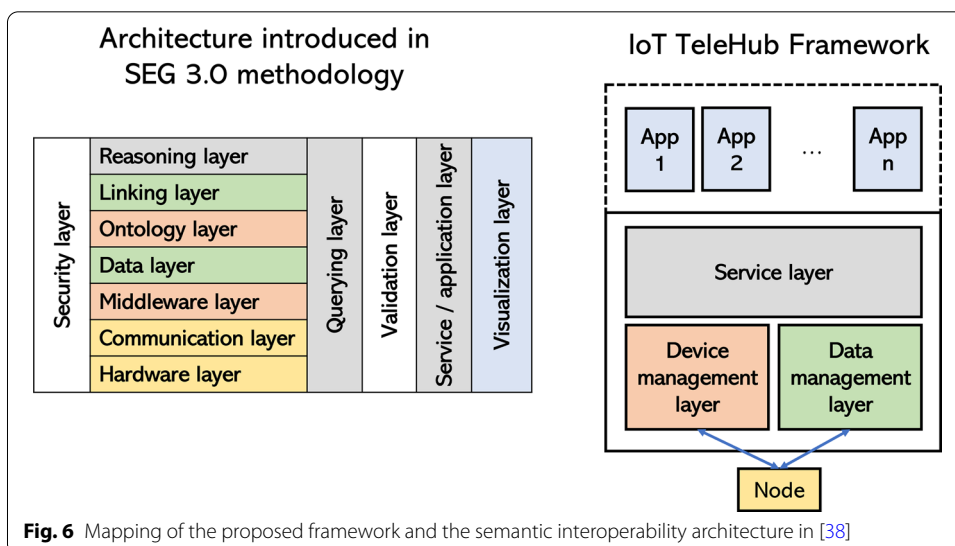


Fig. 6 Mapping of the proposed framework and the semantic interoperability architecture in [38]

particular, we would like to highlight a unique point of our framework through the mapping. That is, while the framework defines the layers and data formats specifically for the case of telemetry data, it is still general enough to be mapped to multiple other frameworks or platforms that cover a wide range of use cases. Another unique point of the framework lies in how we consider and incorporate the machine-readable and machine-understandable aspect in this work, which is detailed further in the [Discussion](#) section.

Figure 6 depicts the 12 layers architecture introduced in the semantic interoperability architecture by Gyrard et al. [38] and its mapping against our *framework*. This architecture provides an interesting comparison due to the number of its fine-grained layering of functionalities, as opposed to the three main layers defined in our core framework, IoT TeleHub. While it is clear that the fine-grained layers defined in [38] provide much better separation of IoT platform functionalities, some smaller platforms might not necessarily need such details. Those platforms might better adopt simpler reference architecture with fewer layers like IoT TeleHub. By mapping our framework to the architecture introduced by Gyrard et al. [38], we would like to show that IoT TeleHub with fewer layers can cover parts of or all of the 12 layers defined in [38]. Hence, platforms that reference our framework can evolve their functional layers as the platforms grow, with sufficiently clear distinction between layers.

The device management and data management layers in our framework are equal, thus might be mapped in no specific order. We describe the mapping as follows.

- *Hardware and Communication layer*. Responsible for getting data from IoT devices and sending them to the Internet. Maps to our Node as the physical IoT device.
- *Middleware and Ontology layer*. Responsible for harmonizing existing platforms and defining unified data models. Maps to the device management layer in IoT TeleHub. The heterogeneity of devices or platforms as data providers can be abstracted into virtual nodes.
- *Data and Linking layer*. Consolidates and enriches data from heterogeneous sources. Maps to the data management layer which serves as the “data channel” in IoT TeleHub. The data enrichment is actually defined in *virtual node representation* in the device management layer. However, as the data management layer also works with the *virtual node representation* during data consolidation, we map the enrichment functionality to this layer as well.
- *Reasoning and Querying layer*. Responsible for selecting data and, if required, deducing information from the data. Maps to the service layer that provides access to data in our framework. The service layer could provide various data formats to upper layers that may involve just data queries or even some data processing.
- *Security layer*. Maps to every layer in our framework as each functionality requires security. For instance, the data management layer in IoT TeleHub performs node authentication before accepting data from the sending parties, while the device management layer provides abstraction of the actual hardware types.
- *Validation layer*. Maps to every layer in our framework as the layers are dependent on semantics described by other layers.
- *Service layer*. Straightforwardly mapped to the service layer in our framework that provides data access to upper layers (e.g. Application layer) or other platforms.

- *Visualization layer*. It provides user interfaces and maps to the Application in our framework.

The architecture for IoT device virtualization proposed in [42] discussed an interesting notion of effectively sharing IoT devices between applications, and introduced a minimal yet functional architecture along with its deployment in a microservices-driven setup. As opposed to the previous mapping to [38], in this mapping we instead expand the layers in Ogawa et al. 's architecture. This shows that our framework is mapped properly to a minimal device sharing architecture that has been proven to work. Furthermore, we expand the definition of IoT devices sharing on a cross-platform basis. That is, our framework provides a reference for individual IoT platforms that also accommodates cross-platform sharing. This mimics the typical real world use case where different institutions and organizations have their own platforms, but require interoperability with other platforms.

Our *framework* maps straightforwardly to the IoT device virtualization architecture [42], as shown in Fig. 7. Sensors and Applications map to *Nodes* and *Applications* in our framework, respectively. Function components that provide common APIs also map directly to our *service layer*. Lastly, the Gateway component that performs data pooling functionality is represented by our framework's *device management layer* and *data management layer*.

Lastly, we map our *framework* to a reference IoT architecture presented by Guth, et al. [49]. While they do not explicitly discuss IoT platforms for data sharing, it provides a general architecture that most IoT platforms adhere to. We show through this mapping that our data sharing semantic can be adopted to existing platforms, regardless of the platforms having data sharing capability or not. The IoT integration middleware in the reference architecture is described broadly and covers most of the functionalities in IoT platforms. Thus our core *framework* maps directly to it, as shown in Fig. 8. We deduce that our *framework* is general in the sense that it fits into the IoT integration middleware.

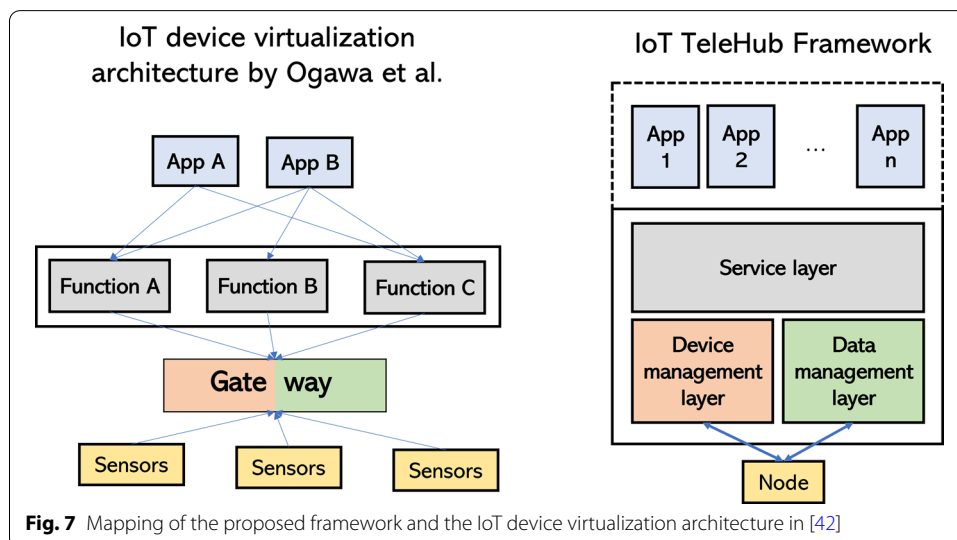
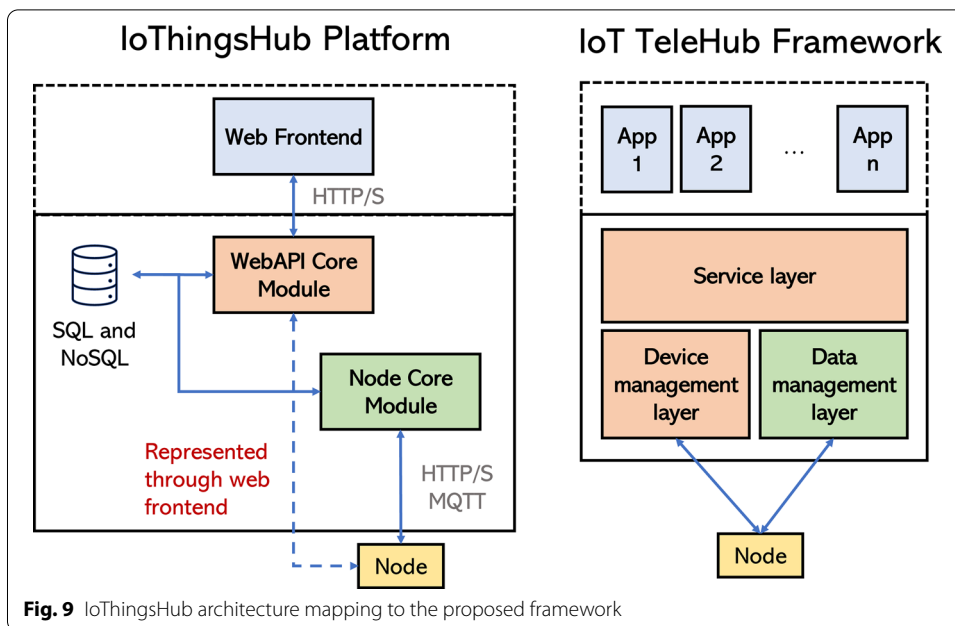
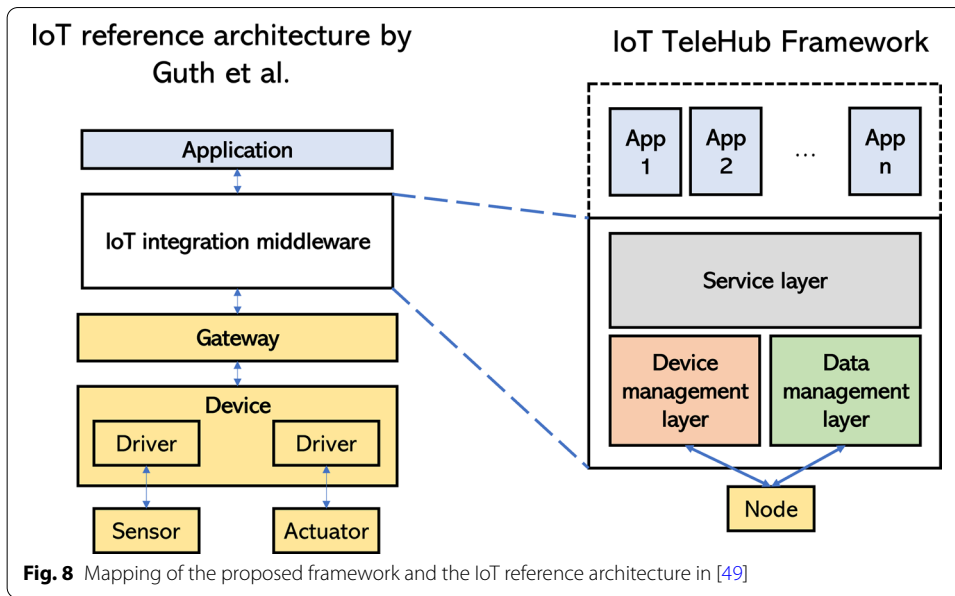


Fig. 7 Mapping of the proposed framework and the IoT device virtualization architecture in [42]



However, more detailed work is required to incorporate the data sharing semantic into each platform rather than mapping it to the reference architecture.

Prototype as an implementation of the framework

In this section, we present IoThingsHub, an IoT platform built for real-time data sharing. We discuss IoThingsHub architectures and show how we implement IoThingsHub based on our *framework* with existing web technologies. Lastly, we briefly present IoThingsHub features and its data sharing function.

IoThingsHub architecture

Figure 9 depicts IoThingsHub architecture on the left and our *framework* on the right side. The mapping between elements is shown with color-coding, i.e., elements with the same color. IoThingsHub utilizes a NoSQL database to keep the telemetry data and SQL relational database for the rest, detailed in the next subsection. Precisely, IoThingsHub comprises of four main elements:

1. *Node*. A typical IoT device that is intended to connect to the platform.
2. *Web API Core Module*. This module covers the *device management layer* and the *service layer* functionalities. It provides HTTP API to the Web Frontend. The figure shows that users represent the Node as a virtual node via the Web Frontend to access the Web API Core Module. We still consider the direct logical connection between the Node and the Web API Core Module as Web Frontend is just one of the methods to communicate with the Web API Core Module.
3. *Node Core Module*. This module covers the *data management layer* functionality. It listens on HTTP and MQTT connections. Nodes can send telemetry data to the Node Core Module after they are virtually represented in the platform. Nodes must always attach their access tokens with the telemetry data for authentication. The Node Core Module will store the telemetry data according to the credentials in the access token.
4. *Web Frontend*. The application that connects to the *service layer*. We will discuss this next in IoThingsHub features.

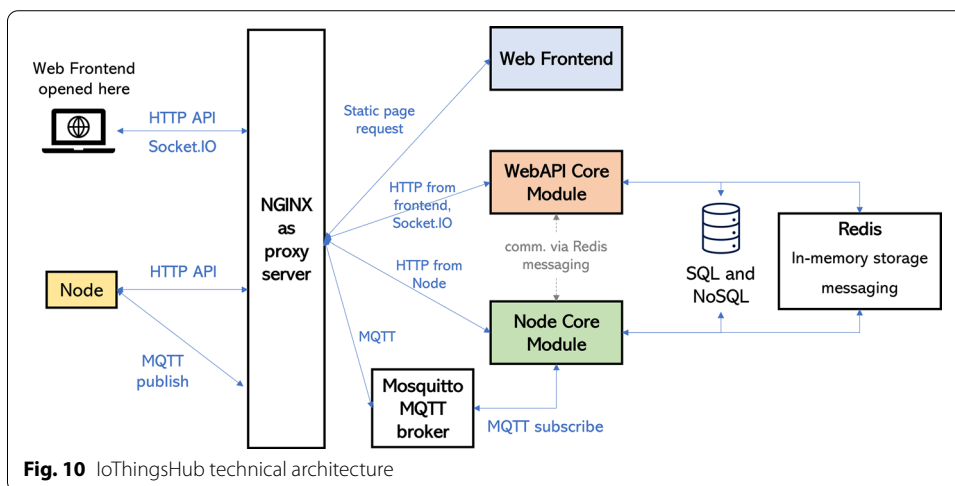


Figure 10 describes the IoThingsHub platform implementation in more detail. We used these technologies to build the platform:

- MySQL Community Server v8.0 for the relational database, and MongoDB v4.4 for the NoSQL database. We also used Redis v5 for its in-memory storage and messaging functions.
- NodeJS to build the WebAPI Core Module and the Node Core Module as we characterize that our platform operations are IO-heavy, and the asynchronous nature of NodeJS fits that. We used the ExpressJS framework for building the API endpoints.
- Socket IO for real-time communication between the WebAPI Core Module and the Web frontend (only for real-time widgets) running on web browsers.
- Mosquitto MQTT broker for MQTT communication between the Nodes and the Node Core Module.
- NGINX as the proxy server, routing all traffic into the internal elements. Those endpoints are the WebAPI Core HTTP endpoints, the Node Core HTTP endpoints, the routing of MQTT streams into Mosquitto MQTT broker, the Web frontend static files, and the Socket IO endpoints.

Based on the detailed architecture, we describe further the technical implementation of these IoThingsHub's elements as follows.

- *The WebAPI Core Module* connects with both MySQL and MongoDB, and listens to HTTP API requests from the Web frontend. It utilizes Redis in-memory storage to keep a list of active Web frontend clients that has 'Dashboard' pages with real-time Widgets open. This list will be used for propagating real-time telemetry updates to the Web frontend via Socket IO. The real-time update to the Web frontend does not affect the data logging process, i.e., the telemetry data will still be stored in MongoDB whether the Web frontend clients list is empty or not.
- *The Node Core Module* connects with both MySQL and MongoDB, and accepts telemetry data sent via HTTP from the Nodes. It also subscribes to the Mosquitto broker on selected channels, listening to telemetry data from the Nodes. For each telemetry data received, the Node Core Module saves it to MongoDB accordingly and checks in Redis in-memory storage whether there are any active Web frontend clients waiting for real-time data. If so, the Node Core Module sends the information to the WebAPI Core Module via Redis messaging service, which then propagates the data to the corresponding Web frontend clients.

SQL and NoSQL in IoThingsHub

In the implementation, we used NoSQL only for nodes' telemetry data because we needed the flexibility in storing those telemetry data. For instance, a user wants to store temperature and humidity data, while another user intends to store luminance, ambient noise, and humidity data. The widely varying type and number of 'data fields' that each user and each node requires led to NoSQL as the storage solution. However, adhering to the data storage format in Fig. 3, we kept some rigidity enforced via the data

management layer. That is, the only flexible part in the NoSQL data format is the list of ‘data fields’. The content of those data fields are rigidly formatted as an array of timestamp-value pairs.

The SQL stores information for Users, Projects, Nodes, and Widgets. Particularly, the Nodes information contains the node unique ID, the ownership (which user owns the node), and the ‘data fields’. We enforce the rigidity of the SQL on the NoSQL database in this manner:

- For each node registered to the system, the *device management layer* creates a new entry in the SQL database, and instructs the *data management layer* to create a corresponding entry in the NoSQL database, sharing the same node ID and ‘data fields’.
- Each Widget entry in the SQL database stores information of which nodes and which ‘data fields’ that it will display. When querying data for the Widget, the *service layer* will query the NoSQL database based on the information in the Widget. Hence, the actual IoT node must send their telemetry data in accordance with the SQL database, else their data will not be shown in the Widgets (hence inaccessible).
- For instance, a node is registered with two data fields, ‘luminance’ and ‘noise’. A Widget is created to show those two fields as a time series data. The actual IoT node should send its telemetry data marked as ‘luminance’ and ‘noise’, else it will not be stored in the NoSQL, or even displayed in the Widget.

IoThingsHub features

Users connect their IoT nodes to IoThingsHub and view their data in a customizable dashboard. We explain IoThingsHub use cases and features with these four entities in the platform:

1. *User*. A typical user in a system that is allowed to login and access IoThingsHub. A user may own multiple Nodes and multiple Projects.

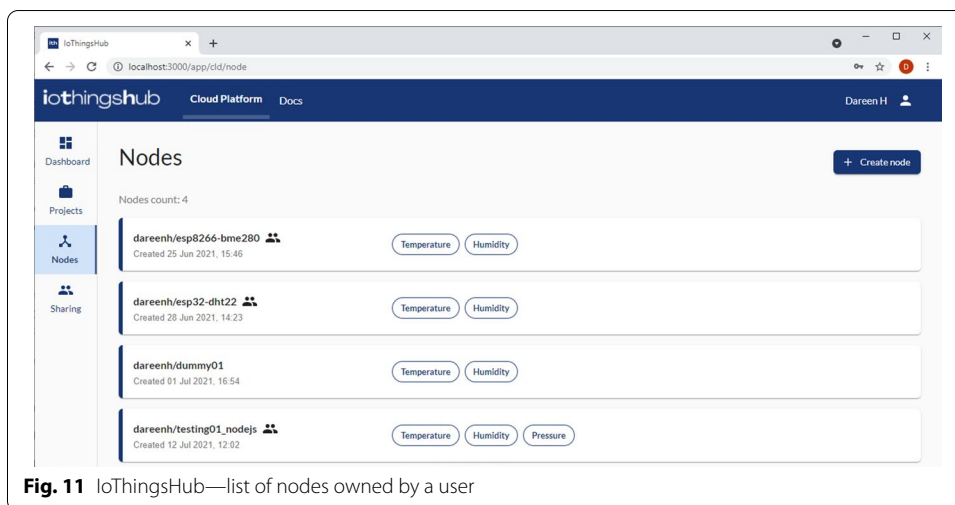


Fig. 11 IoThingsHub—list of nodes owned by a user

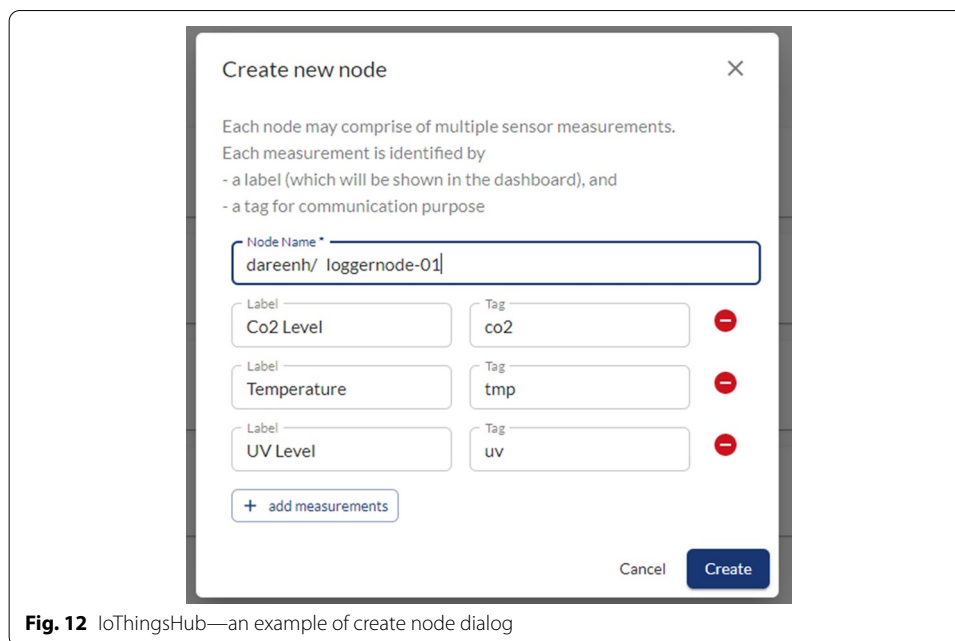


Fig. 12 IoThingsHub—an example of create node dialog

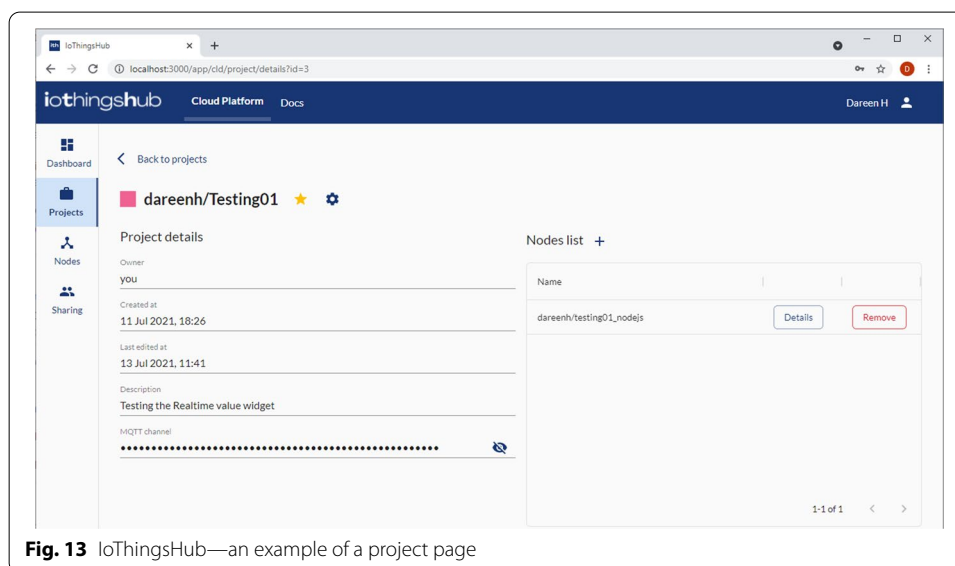


Fig. 13 IoThingsHub—an example of a project page

2. *Node*. A virtual representation of IoT nodes. It follows the *virtual node format* discussed in the framework. Users can access the nodes they own as shown in Fig. 11. Note that there is a “group icon” on Nodes with public discoverability. Figure 12 shows a dialog for creating a new virtual node representation in IoThingsHub.
3. *Project*. A virtual representation for an IoT system. Nodes can be added under one or more Projects, where their data is accessible for the Project. Each Project is assigned a dashboard that holds multiple Widgets and assigned a secret MQTT channel for nodes to send their data. Figure 13 shows an example of a Project page containing the project details and list of nodes added under that Project. As in the Node, Project name is also uniquely namespaced with the username.

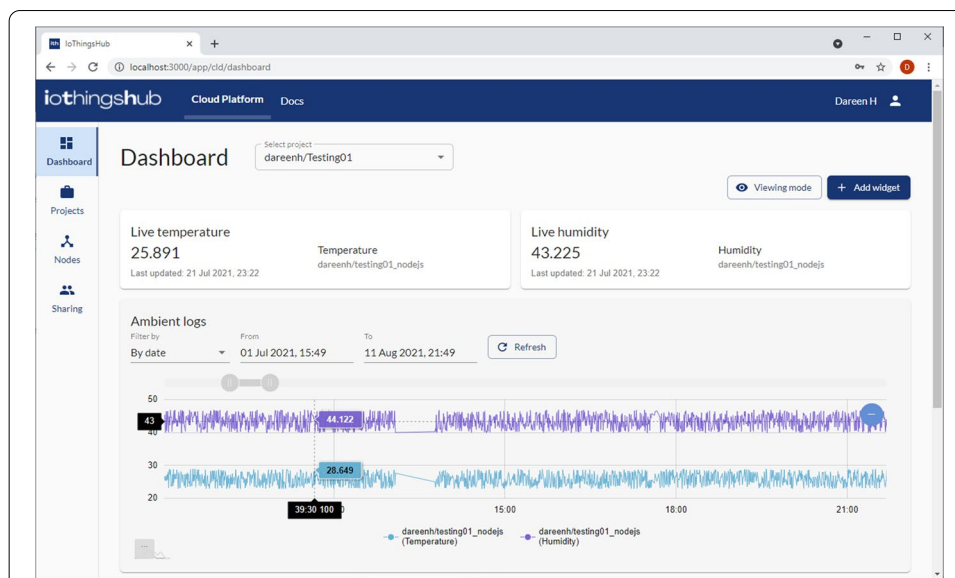


Fig. 14 IoTThingsHub—an example of a project's dashboard page

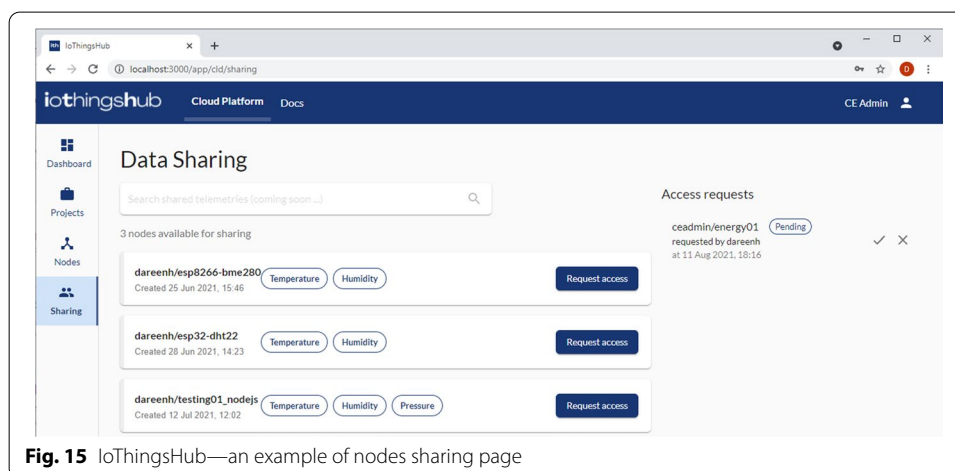


Fig. 15 IoTThingsHub—an example of nodes sharing page

4. *Widget*. A typical IoT widget for displaying nodes data in various forms. Users may add, modify, and remove Widgets to their interests. Per the writing of this paper, there are three widget types available in IoTThingsHub; (1) *time series line graph*, (2) *real-time line graph*, and (3) *real-time value*. Figure 14 displays an example of a Project's dashboard, containing two *real-time value* Widgets and one *time series line graph* Widget.

Users can browse other user's public nodes via the *Sharing* menu and request access to any node of interest, as shown in Fig. 15. Users can also view a list of access requests from other users and either approve or deny the requests. In the example shown in Fig. 15, the user discovers three public nodes from other users and receives

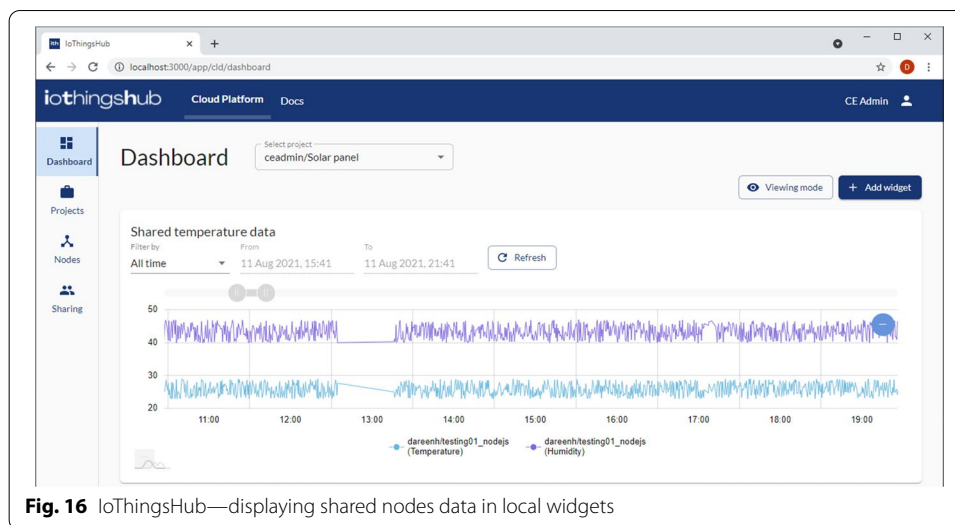


Fig. 16 IoTThingsHub—displaying shared nodes data in local widgets

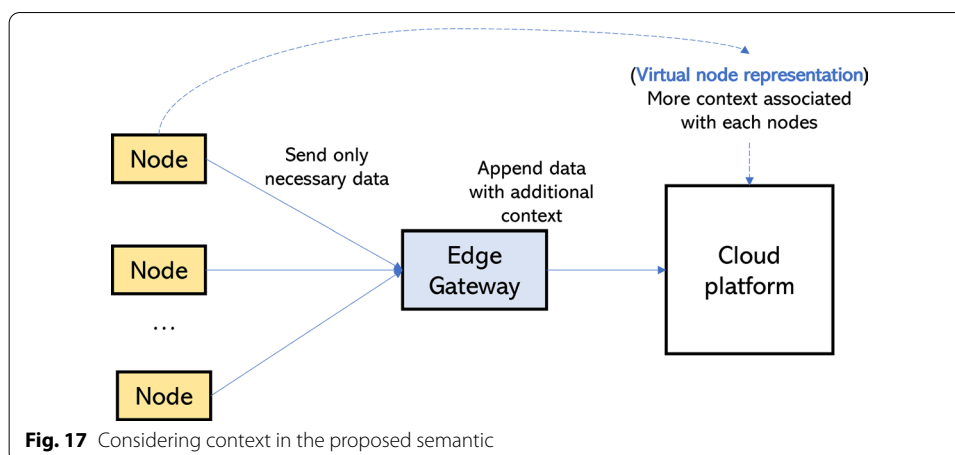
one access request for the node named “ceadmin/energy01” that he or she owns. After receiving approval from the node owner, the requesting user may use the requested node’s data in their project dashboard. Figure 16 shows an example of a user named “CE Admin” displaying the data from a shared node named “dareenh/testing01_nodejs”. It displays the data in the *time series line graph* Widget, which is entirely local to “CE Admin”.

Discussion

This section highlights the main idea that underlies our proposed framework and semantics. That is, how we view and consider the machine-readable and machine-understandable notions in this work, which is defined specifically in the domain of physical-world telemetry data over IoT. We based our framework and semantics on the concept that physical-world data points provide meaningful information as a group, which at its simplest form is a time series.

We consider the machine-readable notion in the sense that a machine-readable (syntactic) format should be as compact and as natural as possible for the machine to process and transceive. The compactness aspect is reflected in the use of labels and tags to separate the human-readable and machine-readable format as shown in Fig. 2. As depicted in Fig. 3, the data storage format advocates storing measurements as separated arrays of timestamp-value pairs. The array data structure comes naturally as we perceive telemetry data as time series, and it is a simple yet versatile data structure that can be manipulated freely and quickly by computers.

Similarly, we consider the machine-understandable aspect in how to incorporate context into the device and data-related semantics in a flexible way, without compromising the performance. This is represented in the virtual node representation and the data storage format, most notably by the use of labels and tags. As shown in Fig. 3, both node representation and data storage formats allow addition of context information flexibly without affecting their basic functionality. For example, in the virtual node representation, we can add fields regarding location, ownership, device characteristics (e.g.,



remote or stationed) in the same level as the node ID field. Going further, the measurement fields can be extended by measurement unit, method, and collection period fields as discussed by Zeng et al. [40]. All this contextual information is separated from the measurement data that the nodes must send, allowing bandwidth saving. If required, contextual information can also be added into each telemetry data point without burdening the node. Consider the case of a fleet of data loggers on solar panel units sending measurements to an edge gateway that communicates with a central server. The data loggers send only necessary measurements to the gateway, which then append the data with contextual information such as weather conditions before sending it to the central server. This idea is illustrated in Fig. 17.

Conclusion , limitations and opportunities for future research

In this work, we proposed IoT Telemetry Data Hub (IoT TeleHub), a general IoT framework and semantic for telemetry data collection and sharing. We presented three core layers that constitute the framework by defining their functionality and respective data semantics, i.e., *device management layer*, *data management layer*, and *service layer*. For interoperability with existing IoT platforms, the defined data semantics only covers parts related to telemetry data sharing. We characterized telemetry data as time-series data and defined the respective data storage semantic. IoT devices are abstracted as *virtual nodes* in our framework to provide separation of hardware and software.

We demonstrated two unique points of our work. First, the framework's generality by mapping the framework to several existing IoT platforms. It shows that our framework can map well to existing platforms if performed on a one-on-one basis. Second, we designed the framework while considering the machine-readable and machine-understandable notion in regard to the performance aspect of resource-constrained IoT devices, e.g., bandwidth, battery, and computational power.

Lastly, we presented IoThingsHub, an IoT platform for real-time data sharing, developed based on the presented framework. We built the platform with existing technologies such as HTTP, MQTT, SQL, NoSQL. It demonstrates the data sharing

service with a *nodes-users* relational table that is detached from the data collection and storage.

Future works may extend the framework and semantic by adding context information and various metadata into the *virtual node representation*. These extra fields allow better understanding and indexing of the nodes and data shared in the platform. Furthermore, we can employ more fine-grained sharing by extending the *nodes-users* relational table with a list of measurements.

Abbreviations

API: Application Programming Interface; IoT: Internet of Things; MQTT: MQ Telemetry Transport; TSDB: Time Series Database; W3C: World Wide Web Consortium.

Acknowledgements

The authors would like to thank you Universitas Multimedia Nusantara for supporting this work, in particular for the infrastructure required for the implementation.

Authors' contributions

DH has initiated and made the first draft of this paper. Both authors read and approved the final manuscript.

Funding

Not applicable.

Availability of data and materials

Data generated during the study by the platform operation are available from the corresponding author on a reasonable request. The platform that implements the *framework* is hosted under the university's domain, and available internally.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Received: 28 August 2021 Accepted: 10 December 2021

Published online: 04 January 2022

References

1. L'Heureux A, Grolinger K, Elyamany HF, Capretz MAM. Machine learning with big data: challenges and approaches. *IEEE Access*. 2017;5:7776–97. <https://doi.org/10.1109/ACCESS.2017.2696365>.
2. Tsai C-W, Lai C-F, Chao H-C, Vasilakos AV. Big data analytics: a survey. *J Big Data*. 2015;2(1):21. <https://doi.org/10.1186/s40537-015-0030-3>.
3. Kubina M, Varmus M, Kubinova I Use of big data for competitive advantage of company. *Procedia Economics and Finance* 2015;26:561–565. [https://doi.org/10.1016/S2212-5671\(15\)00955-7](https://doi.org/10.1016/S2212-5671(15)00955-7). 4th World Conference on Business, Economics and Management (WCBEM-2015).
4. Statista: Number of Internet of Things (IoT) Connected Devices Worldwide from 2019 to 2030. <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/>.
5. Cisco: Cisco Annual Internet Report (2018–2023) White Paper. <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>.
6. Statista: Data Volume of Internet of Things (IoT) Connections Worldwide in 2019 and 2025. <https://www.statista.com/statistics/1017863/worldwide-iot-connected-devices-data-size/>.
7. Morais CMD, Sadok D, Kelner J. An IoT sensor and scenario survey for data researchers. *J Brazil Comput Soc*. 2019;25(1):4. <https://doi.org/10.1186/s13173-019-0085-7>.
8. Hossain E, Khan I, Un-Noor F, Sikander SS, Sunny MSH. Application of big data and machine learning in smart grid, and associated security concerns: a review. *IEEE Access*. 2019;7:13960–88. <https://doi.org/10.1109/ACCESS.2019.2894819>.
9. Shapi MKM, Ramli NA, Awal LJ. Energy consumption prediction by using machine learning for smart building: case study in Malaysia. *Dev Built Environ*. 2021;5:100037. <https://doi.org/10.1016/j.dibe.2020.100037>.
10. Zhang Y, Huang T, Bompard EF. Big data analytics in smart grids: a review. *Energy Inform*. 2018;1(1):8. <https://doi.org/10.1186/s42162-018-0007-5>.
11. Seethalakshmi P, Venkatalakshmi K. Prediction of energy demand in smart grid using deep neural networks with optimizer ensembles. In: 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), 2020:pp. 1–5. <https://doi.org/10.1109/ICCMC48092.2020.ICCMC-000109>.

12. Karyono K, Abdullah BM, Cotgrave AJ, Bras A. The adaptive thermal comfort review from the 1920s, the present, and the future. *Dev Built Environ*. 2020;4:100032. <https://doi.org/10.1016/j.dibe.2020.100032>.
13. Tushar W, Wijerathne N, Li W-T, Yuen C, Poor HV, Saha TK, Wood KL. Internet of things for green building management: disruptive innovations through low-cost sensor technology and artificial intelligence. *IEEE Signal Process Mag*. 2018;35(5):100–10. <https://doi.org/10.1109/MSP.2018.2842096>.
14. Alawadi S, Mera D, Fernández-Delgado M, Alkhabbas F, Olsson CM, Davidsson P. A comparison of machine learning algorithms for forecasting indoor temperature in smart buildings. *Energy Syst*. 2020. <https://doi.org/10.1007/s12667-020-00376-x>.
15. Rusli A, Halim DK Towards an integrated hybrid mobile application for smart campus using location-based smart notification. In: 2019 International Conference on Engineering, Science, and Industrial Applications (ICESI), 2019:pp. 1–6. <https://doi.org/10.1109/ICESI.2019.8863022>.
16. Halim D, Rusli A. Wi-fi based indoor localization for location-based smart notification. *IJNMT (International Journal of New Media Technology)*. 2020;7(1):43–50. <https://doi.org/10.31937/ijnmt.v7i1.1628>.
17. Sun D, Wei E, Ma Z, Wu C, Xu S. Optimized cnns to indoor localization through ble sensors using improved pso. *Sensors*. 2021. <https://doi.org/10.3390/s21061995>.
18. Sadowski S, Spachos P. Rssi-based indoor localization with the internet of things. *IEEE Access*. 2018. <https://doi.org/10.1109/ACCESS.2018.2843325>.
19. Stefanou A-M, Woloszyn M, Jay A, Wurtz E, Buhé, C.: A methodology to assess the ambient temperature of a building using a limited number of sensors. *Energy Procedia* 78, 1944–1949. 6th International Building Physics Conference. IBPC. 2015;2015. <https://doi.org/10.1016/j.egypro.2015.11.377>.
20. Yan X, Liu C, Li M, Hou A, Fan K, Meng Q. Climate compensation and indoor temperature optimal measuring point energy saving control in vav air-conditioning system. *Energies*. 2019. <https://doi.org/10.3390/en12224398>.
21. Popović T, Latinović N, Pečić A, Zečević Zarko, Krstajić B, Djukanović S. Architecting an IoT-enabled platform for precision agriculture and ecological monitoring: a case study. *Comput Electr Agric*. 2017;140:255–65. <https://doi.org/10.1016/j.compag.2017.06.008>.
22. Kassal P, Steinberg MD, Steinberg IM. Wireless chemical sensors and biosensors: a review. *Sens Actuators B Chem*. 2018;266:228–45. <https://doi.org/10.1016/j.snb.2018.03.074>.
23. Villa-Henriksen A, Edwards GTC, Pesonen LA, Green O, Sørensen CAG. Internet of things in arable farming: implementation, applications, challenges and potential. *Biosyst Eng*. 2020;191:60–84. <https://doi.org/10.1016/j.biosystemseng.2019.12.013>.
24. Gopinath R, Kumar M, Prakash Chandra Joshua C, Srinivas K. Energy management using non-intrusive load monitoring techniques—state-of-the-art and future research directions. *Sustain Cities Soc*. 2020;62:102411. <https://doi.org/10.1016/j.scs.2020.102411>.
25. Benedá T, Manera LT. Non-intrusive and intrusive energy monitoring methods overview and their relation with household appliances state sensors devices. In: Iano, Y., Arthur, R., Saotome, O., Vieira Estrela, V., Loschi, H.J. (eds) Proceedings of the 4th Brazilian Technology Symposium (BTSym'18), 2019:pp. 407–415. Springer, Cham.
26. Mumtaz S, Alsoghaily A, Pang Z, Rayes A, Tsang KF, Rodriguez J. Massive internet of things for industrial applications: addressing wireless iiot connectivity challenges and ecosystem fragmentation. *IEEE Ind Electr Mag*. 2017;11(1):28–33. <https://doi.org/10.1109/MIE.2016.2618724>.
27. Khan WZ, Rehman MH, Zangoti HM, Afzal MK, Armi N, Salah K. Industrial internet of things: recent advances, enabling technologies and open challenges. *Comput Electr Eng*. 2020;81:106522. <https://doi.org/10.1016/j.compeleceng.2019.106522>.
28. Moore SJ, Nugent CD, Zhang S, Cleland I. IoT reliability: a review leading to 5 key research directions. *CCF Trans Pervasive Comput Interact*. 2020;2(3):147–63. <https://doi.org/10.1007/s42486-020-00037-z>.
29. Karkouch A, Mousannif H, Moatassime HA, Noel T. A model-driven architecture-based data quality management framework for the internet of things. In: 2016 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech), 2016:pp. 252–259. <https://doi.org/10.1109/CloudTech.2016.7847707>.
30. Fekade B, Maksymyuk T, Kyryk M, Jo M. Probabilistic recovery of incomplete sensed data in IoT. *IEEE Internet Things J*. 2018;5(4):2282–92. <https://doi.org/10.1109/JIOT.2017.2730360>.
31. Azimi I, Pahikkala T, Rahmani AM, Niela-Vilén H, Axelin A, Liljeberg P. Missing data resilient decision-making for healthcare IoT through personalization: a case study on maternal health. *Future Gener Comput Syst*. 2019;96:297–308. <https://doi.org/10.1016/j.future.2019.02.015>.
32. Liu Y, Dillon T, Yu W, Rahayu W, Mostafa F. Missing value imputation for industrial IoT sensor data with large gaps. *IEEE Internet Things J*. 2020;7(8):6855–67. <https://doi.org/10.1109/JIOT.2020.2970467>.
33. Khan SI, Hoque ASML. Sice: an improved missing data imputation technique. *J Big Data*. 2020;7(1):37. <https://doi.org/10.1186/s40537-020-00313-w>.
34. Osman MS, Abu-Mahfouz AM, Page PR. A survey on data imputation techniques: water distribution system as a use case. *IEEE Access*. 2018;6:63279–91. <https://doi.org/10.1109/ACCESS.2018.2877269>.
35. Noura M, Atiquzzaman M, Gaedke M. Interoperability in internet of things: taxonomies and open challenges. *Mobile networks and applications*. 2019;24(3):796–809. <https://doi.org/10.1007/s11036-018-1089-9>.
36. W3C: Semantic Sensor Network Ontology. <https://www.w3.org/TR/vocab-ssn/>.
37. Agarwal R, Fernandez DG, Elsaleh T, Gyrard A, Lanza J, Sanchez L, Georgantas N, Issary V. Unified iot ontology to enable interoperability and federation of testbeds. In: 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), 2016:pp. 70–75. <https://doi.org/10.1109/WF-IoT.2016.7845470>.
38. Gyrard A, Serrano M. Connected smart cities: Interoperability with seg 3.0 for the internet of things. In: 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2016:pp. 796–802. <https://doi.org/10.1109/WAINA.2016.151>.
39. Terziyan V, Kaykova O, Zhovtobryukh D. Ubiroad: Semantic middleware for context-aware smart road environments. In: 2010 Fifth International Conference on Internet and Web Applications and Services, 2010:pp. 295–302. <https://doi.org/10.1109/ICIW.2010.50>.

40. Zeng W, Zhang S, Yen I-L, Bastani F. Invited paper: Semantic iot data description and discovery in the iot-edge-fog-cloud infrastructure. In: 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), 2019;pp. 106–10609.<https://doi.org/10.1109/SOSE.2019.00024>.
41. de Matos E, Tiburski RT, Moratelli CR, Johann Filho S, Amaral LA, Ramachandran G, Krishnamachari B, Hessel F. Context information sharing for the internet of things: a survey. *Comput Netw.* 2020;166:106988. <https://doi.org/10.1016/j.comnet.2019.106988>.
42. Ogawa K, Kanai K, Nakamura K, Kanemitsu H, Katto J, Nakazato H. Iot device virtualization for efficient resource utilization in smart city iot platform. In: 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2019;pp. 419–422. <https://doi.org/10.1109/PERCOMW.2019.8730806>.
43. Cheng B, Solmaz G, Cirillo F, Kovacs E, Terasawa K, Kitazawa A. *fogflow*: Easy programming of iot services over cloud and edges for smart cities. *IEEE Internet Things J.* 2018;5(2):696–707. <https://doi.org/10.1109/JIOT.2017.2747214>.
44. Benazzouz Y, Munilla C, Günalp O, Gallissot M, Gürgen L. Sharing user iot devices in the cloud. In: 2014 IEEE World Forum on Internet of Things (WF-IoT), 2014;p. 373–374. <https://doi.org/10.1109/WF-IoT.2014.6803193>.
45. Sekine H, Kanai K, Katto J, Kanemitsu H, Nakazato H. Iot-centric service function chaining/orchestration and its performance validation. In: 2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC), 2021;pp. 1–4. <https://doi.org/10.1109/CCNC49032.2021.9369538>.
46. Samaniego M, Deters R. Management and internet of things. *Procedia Computer Science* 2016;94, 137–143. <https://doi.org/10.1016/j.procs.2016.08.022>. The 11th International Conference on Future Networks and Communications (FNC 2016) / The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016) / Affiliated Workshops.
47. Sigfox: Sigfox Payload. <https://build.sigfox.com/payload>.
48. Zhang WE, Sheng QZ, Mahmood A, Tran DH, Zaib M, Hamad SA, Aljubairy A, Alhazmi AAF, Sagar S, Ma C. The 10 research topics in the internet of things. In: 2020 IEEE 6th International Conference on Collaboration and Internet Computing (CIC), 2020;pp. 34–43. <https://doi.org/10.1109/CIC50333.2020.00015>.
49. Guth J, Breitenbücher U, Falkenthal M, Fremantle P, Kopp O, Leymann F, Reinfurt L. A detailed analysis of iot platform architectures: concepts, similarities, and differences. In: Di Martino B, Li K-C, Yang LT, Esposito A, editors. *Internet of everything: algorithms, methodologies, technologies and perspectives*. Singapore: Springer; 2018. p. 81–101.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)
