MQTT4SSN: An Ontology for the MQTT Message Protocol

Niklas Doerner*, Maria Maleshkova*

Professorship of Data Engineering, Helmut Schmidt University, Holstenhofweg 85, 22043 Hamburg, Germany

Abstract

In the Web of Things community, the SSN/SOSA ontology, a W3C recommendation, has established itself as a foundational framework for representing semantic sensor networks, modeling sensors, observations, and actuations. However, an explicit representation of message protocols is missing, which limits semantic interoperability in machine-to-machine communication scenarios. To close this gap, we propose MQTT4SSN as an extension to SSN/SOSA that semantically models the MQTT protocol. Furthermore, the ontology addresses use cases without SSN/SOSA implementations by exclusively describing the MQTT messaging protocol. MQTT4SSN represents MQTT entities such as brokers, clients, control packets, topics, and payload metadata, linking them to SSN/SOSA concepts to enable end-to-end traceability between sensing semantics and communication semantics. The ontology captures heterogeneous payload formats, encodings, and transport metadata, enabling machine-interpretable description and integration of transmitted content. MQTT4SSN represents an extensible and reusable resource that is accessible according to the FAIR principles and documented as an Ontology Specification Draft.

Ontology: https://doernern.github.io/MQTT4SSNOntology/MQTT4SSN.owl
Documentation: https://doernern.github.io/MQTT4SSNOntology/documentation/index-en.html
WebVOWL: https://doernern.github.io/MQTT4SSNOntology/documentation/webvowl/index.html
OOPS!: https://doernern.github.io/MQTT4SSNOntology/documentation/OOPSevaluation/oopsEval.html
GitHub: https://github.com/doernern/MQTT4SSNOntology
License: CC BY-NC-SA 4.0
DOI: 10.5281/zenodo.16704302

Keywords

MQTT, Ontology Engineering, Sensor Ontology, Semantic Sensor Network, Message Protocol, Smart Sensors

1. Introduction

In the Web of Things (WoT) community, the **SSN/SOSA** ontology, a W3C recommendation, has established itself as a foundational framework for representing semantic sensor networks. However, while **SSN/SOSA** considers modeling physical systems, sensor deployments, and observational and actuation processes, it does not consider machine-to-machine (M2M) communication protocols [1]. In particular, Message Queuing Telemetry Transport (MQTT) is a widely adopted message protocol in industrial and Internet of Things (IoT) settings for its lightweight, publish-subscribe-based architecture [2, 3]. The lack of the transport component motivates extending the **SSN/SOSA** framework with MQTT's main elements, enabling seamless alignment between observation and actuation process descriptions with transport semantics. A further challenge arises from the various payload formats and encodings used in MQTT message protocols. Payloads may contain plain text, comma-separated values, or structured JSON as format, possibly encoded in various character sets [4, 5]. These heterogeneous payloads are not clearly described to semantic systems unless an explicit annotation with metadata that describes their format and structure is missing. Therefore, there is a need to model such payload metadata semantically to enable parsing, interpretation, and integration of the transmitted content. Moreover, MQTT

ISWC 2025 Companion Volume, November 2–6, 2025, Nara, Japan
*Corresponding author.
†These authors contributed equally.
doernern@hsu-hh.de (N. Doerner*); maleshkm@hsu-hh.de (M. Maleshkova*)
https://www.hsu-hh.de/dataeng/en/team/niklas-doerner/ (N. Doerner*);
https://www.hsu-hh.de/dataeng/en/team/maria-maleshkova/ (M. Maleshkova*)
0009-0004-0088-8633 (N. Doerner*); 0000-0003-3458-4748 (M. Maleshkova*)

topics and payload structures are generally syntactically defined and application-specific. However, in the context of semantic sensor networks, topic names and data structures could be derived from the semantics of the observation [5, 6]. For instance, hierarchical topic paths could describe the associated room, sensing device, sensor, observed property, and feature of interest. Likewise, the structure and arrangement of payload content, such as the column order in CSV or key nesting JSON syntax, could describe the made observation collection, for instance, with the associated room, sensor, and feature of interest. The hierarchy of MQTT topic names and payload structures raises the question of whether there is a relationship to the entities defined by **SSN/SOSA**.

To address these challenges, we initially defined the following three research questions (RQs):

- **RQ1:** How can the MQTT message protocol be semantically represented in a way that aligns with an existing sensor network ontology, such as W3C SSN/SOSA?
- **RQ2:** How can heterogeneous payload formats and character encodings be represented in a semantic model to enable machine-interpretable integration of the transmitted data?
- **RQ3:** To what extent can semantic MQTT topic names and structured payload representations be derived from observation-related information, such as the observed feature of interest, the used sensor, and the observed property?

The OASIS standard **MQTT** (Message Queuing Telemetry Transport) is a messaging protocol designed for communication in M2M and IoT contexts [7]. It operates on a publish/subscribe model with four main components: clients, which act as publishers and/or subscribers, and a central broker that manages message routing. Clients publish messages to topics and subscribe to topic filters, which can include wildcards to capture related topic streams dynamically. For communication, MQTT uses control packets, which contain a fixed header in all control packets, and depending on the packet type, an optional variable header and payload. The MQTT specification defines fourteen different types of control packets, managing actions such as connection establishment (`CONNECT`, `CONNACK`), publishing (`PUBLISH`, `PUBACK`), and subscription (`SUBSCRIBE`, `SUBACK`). The payload of the `PUBLISH` control packet contains application data, such as sensor data, and can vary in format and encoding. MQTT delivers Application Messages according to three different Quality of Service (`QoS`) levels to ensure reliability in message delivery. There are additional header flags for transmission, such as the `RETAIN` and `DUP` flags. The MQTT message protocol typically runs over the TCP/IP transport protocol.

Our **MQTT4SSN** ontology focuses on MQTT version 3.1.1[7] to ensure compatibility with established systems and model the foundation. However, we designed the model to remain extensible for future integration of MQTT 5.0 features. The newer MQTT Version 5.0[8] standard extends the protocol with richer metadata, reason codes, user properties, and improved error handling, enhancing flexibility and scalability for modern IoT applications [8]. There exists a draft from the OASIS group, the MQTT For Sensor Networks (MQTT-SN)[9], which adapts MQTT for non-TCP/IP networks such as UDP, Zigbee, or Bluetooth. It simplifies message headers and introduces topic IDs to support constrained devices in wireless sensor networks (WSNs).

The remainder of this paper includes the following: Section 2 provides an overview of related ontologies and data models relevant to MQTT, IoT, and WoT. Section 3 explains the ontology engineering process of **MQTT4SSN**, covering its core concepts 3.1, semantic relations 3.2, and applied design patterns 3.3. Section 4 outlines the evaluation methodology and results, including ontology verification 4.1 and validation 4.2 based on predefined competency questions. Finally, Section 5 summarizes the paper and highlights possible directions for future research.

2. Related Data Models

In the landscape of semantic data modeling for IoT and WoT, numerous ontologies such as the **W3C SSN/SOSA** ontology have been developed that describe sensing and actuation devices, observation procedures, and environments. However, despite the central role of communication protocols in IoT infrastructures, very few of these ontologies explicitly address transport mechanisms such as MQTT.

To the best of our knowledge, the only dedicated effort toward semantic representation of MQTT is the **MQTT to RDF** draft ontology defined in the W3C WoT Binding Templates. This fragmentation has created a gap between the semantic modeling of observation processes and the underlying transport layers that enable real-time M2M communication [10, 6].

The **W3C SSN/SOSA** ontology is the most widely adopted standard for modeling sensors, observations, actuators, and sampling activities in semantic IoT contexts. Originally published as a W3C Recommendation in 2017[11], **SOSA** serves as a lightweight core, while **SSN** adds richer axioms for more expressiveness. A new working draft further extends the model and enriches its interoperability across domains [12]. Despite its complexity, **SSN/SOSA** does not specify how data is transmitted or how protocols like MQTT interact with observation streams, leaving a crucial semantic gap.

The **WoT MQTT to RDF** ontology draft, developed as part of the W3C WoT Binding Templates, provides, to the best of our knowledge, the only formal semantic representation of MQTT, especially the description of the control packet structure, topic filters, and the network entities such as broker and client [13]. This draft models protocol-level aspects of MQTT but neglects the modeling of sensing systems, such as **SSN/SOSA**. Consequently, it cannot directly align MQTT messages with sensor observations, features of interest, or platforms.

The **QUDT** (Quantities, Units, Dimensions, and Data Types) vocabulary is another essential component for semantic IoT modeling. Frequently combined with **SSN/SOSA**, **QUDT** provides a rigorous and extensible vocabulary for expressing physical units, scales, and quantity values [11, 14]. It enables precise representation of observation results (e.g., temperature in degree Celsius, length in millimeters), which is vital for machine interpretation and data integration [15].
Among the broader IoT ontologies, several have attempted to provide domain models for smart environments and device interactions:

- **SAREF** (Smart Applications REFerence ontology), maintained by ETSI, focuses on linking different domains. Its modular structure includes extensions for smart cities, buildings, and industry, and it emphasizes interoperability across domains [16]. However, **SAREF** does not describe communication transport, relying instead on linking to existing data representations.
- **FIESTA-IoT** was developed in the context of IoT testbed federation and emphasizes cross-platform interoperability. It builds heavily on **SSN/SOSA** and adds vocabularies for testbeds, datasets, and experimental metadata [17, 18]. While it provides integration logic, it does not address message protocols like MQTT.
- **IoT-O** is an ontology designed to support semantic interoperability across IoT domains. It models devices, sensors, and platforms and explicitly reuses **SSN/SOSA** [19]. Although rich in device and service descriptions, IoT-O lacks support for protocol semantics and dynamic message exchange.
- **IoT-Lite** is a lightweight ontology designed to enable semantic interoperability in resource-constrained IoT environments. It simplifies the modeling of IoT devices and services to facilitate implementation in low-power or embedded contexts [20]. However, the transport components are neglected here.
- **S3N** (Semantic Smart Sensor Network) builds directly on top of **SSN/SOSA** and WoT Thing Descriptions to describe smart sensors and their capabilities [21]. While it supports modular modeling of sensor components and protocols, its MQTT handling is limited and not aligned with the WoT **MQTT to RDF** effort.

In summary, although various ontologies exist for describing IoT devices and services, they often lack a consistent or integrated model for communication transport. The **MQTT to RDF** ontology from the WoT Binding Templates is, to the best of our knowledge, a unique attempt to model MQTT semantics, but does not interlink with domain models like **SSN/SOSA**. Conversely, **SSN/SOSA** offers mature and extensible semantics for observations but is agnostic to how these observations are communicated.

3. Ontology Development

In this work, we propose the **MQTT4SSN** extension, an ontology developed to provide a semantic representation of the MQTT message protocol, including the structure of transmitted data. It extends the well-established **SSN/SOSA** ontology by introducing MQTT-specific components such as the network entities broker and client, control packet structures with their payloads, and their associated topics. The ontology was designed to model automated process data captured and transmitted via MQTT packets in the context of the EKI[1] project. The development was guided by initially defined research questions (RQs), a comprehensive set of competency questions (CQs), and a review of existing ontology design patterns (ODPs). These elements shaped the ontology's structure during design and implementation and served as the basis for its evaluation.

Following best practice in ontology engineering, we analyzed existing ODPs regarding reusability and interoperability. The W3C Web of Things (WoT) MQTT Binding Ontology draft, hereafter referred to as **MQV**, was considered as a potential ODP. We used this ontology draft as a starting point for our ontology design and adapted most of the **MQV** concepts into our ontology. Thereafter, we extend it with further elements to fit our requirements on the MQTT side fully, but also enable the semantic relation to **SSN/SOSA**. Special attention was paid to the representation of heterogeneous payload encoding, enabling the ontology to handle structured data such as CSV or JSON. Furthermore, we integrated the QUDT vocabulary in the traditional **SSN/SOSA** context and the **MQTT4SSN** extension.

To ensure that the ontology is Findable, Accessible, Interoperable, and Reusable (FAIR), **MQTT4SSN** is made openly available on GitHub and permanently archived via Zenodo under a DOI, with complete access to all files. We provide the ontology under an open license and richly annotated with labels, comments, metadata, and documentation to promote adoption and reuse by the semantic web and engineering community.

3.1. Core Concepts

From the outset, we designed **MQTT4SSN**, an extension of the well-established **SSN/SOSA** ontology by incorporating a semantic representation of the MQTT message protocol. While **SSN/SOSA** provides a solid foundation for modeling sensing systems, including sensors and actuators and their linkage to observations, properties, and features of interest, **MQTT4SSN** complements this by explicitly modeling the structure and metadata of the transmitted data. The key elements of MQTT, which are the network entities broker and client, the different control packets with their payload, and their associated topics, form the core of the ontology, along with their interrelations.

This integration enables seamless alignment between observation, actuation, and sampling semantics and M2M communication semantics, thereby supporting end-to-end traceability and semantic interoperability across distributed IoT systems.

To describe payloads in a meaningful and machine-interpretable way, the ontology includes properties for specifying the payload content type (MIME type) and character encoding. In addition, **MQTT4SSN** considered transport-specific characteristics such as quality of service levels, retain and DUP flags to allow for the traceability of message delivery. Despite the incompleteness of the **MQV** draft ontology, we adapted core concepts in **MQTT4SSN** and established partial semantic mappings to align both models for future interoperability.

3.2. Semantic Relations

At the core of this model are four major class clusters: the network infrastructure, the control packet hierarchy, the topic with its relations to different payloads with the relation to **SSN/SOSA**, and the topic subject alignment with **SSN/SOSA** sensing semantics.

Within the network infrastructure, depicted in figure 1, the ontology defines the entities client and the broker as subtypes of network participants involved as actors in MQTT communication. A

[1] https://dtecbw.de/home/forschung/hsu/projekt-eki/, accessed: September 12, 2025

broker has a host address and a port number, while a client has a client ID. The network participants' interaction is mediated by the network connection class, which can be optionally encrypted with TLS and is characterized by initiation and acceptance relations: the connection is initiated by the client and accepted by the broker. The client also explicitly maintains a connection to a specific broker, forming the foundation for message exchange.

Figure 1: Network Infrastructure relations

The Communication is described through the control packet hierarchy, depicted in figure 2. The publish, subscribe, and unsubscribe packet types are modeled as subclasses of a general control packet class. These packets contain structured fixed and variable headers as well as payloads. For example (seen in figures 2, 3), a publish packet includes a fixed header, defining a quality of service level, retain and DUP flag, a variable header which typically includes the topic, and a publish payload that carries encoded observation data as content with certain content type, character encoding and perhaps a content delimiter. Subscribe and unsubscribe packets follow a similar structural breakdown, with specific headers and payload compositions. A variable header can have a packet identifier. All payloads have a content-dependent size. Clients are linked to the control packets they send, while brokers can receive and forward them, enabling the messaging pipeline, as seen in figure 1.

Topics and topic filters represent another essential part of MQTT communication, depicted in figure 3. Sensors and actuators that a client hosts can observe or listen to topics. Topic filters, used during subscription and unsubscription, specify patterns for selecting specific topics. The ontology captures the relationship between topic filters and matched topics and supports modeling individual subscription entries with specific quality of service levels. These filters are linked with the subscription payload over the subscription entry or the unsubscription payload.

One goal of the **MQTT4SSN** ontology was the extension of **SSN/SOSA**, shown in figure 3. The client class runs on a platform for hosting **SOSA** sensors or actuators, thus bridging MQTT infrastructure and the physical sensing world. A sensor hosted by a client may publish data to a topic, while an actuator may listen to a topic for actuation commands. On the **SSN/SOSA** side, both systems are hosted by a platform, and on the MQTT side by a client. Another alignment shows figure 4, depicting a topic that

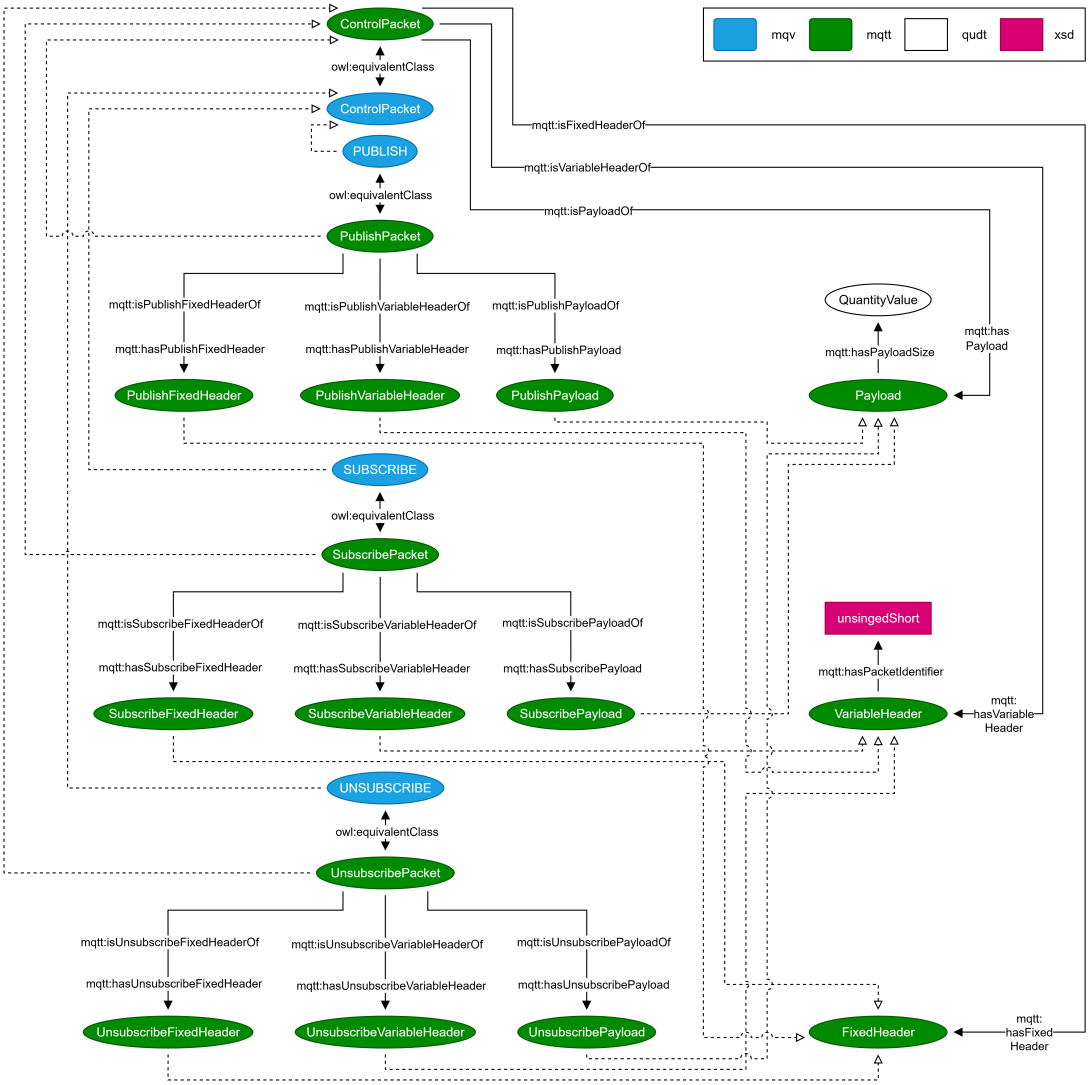

Figure 2: Control Packet hierarchy

can be related **to one or more topic subjects, represented by SSN/SOSA** entities. An observation or actuation (possibly as a collection) can describe a topic name with an associated feature of interest and property.

Further, the ontology relates MQTT topics and payloads to **SOSA** observations, also depicted in figure 3. Each topic may describe a specific observation or a collection of observations, which are modeled as being described by topics. Payloads of publish packets can encode single observations or actuation collections of them.

Beyond the specification of conceptual axioms, logical constructs such as disjointness were incorporated to enforce explicit class separation, while inverse relationships were introduced to define predicates in a bidirectional manner, thereby enhancing semantic rigor and ensuring consistency across the ontology.

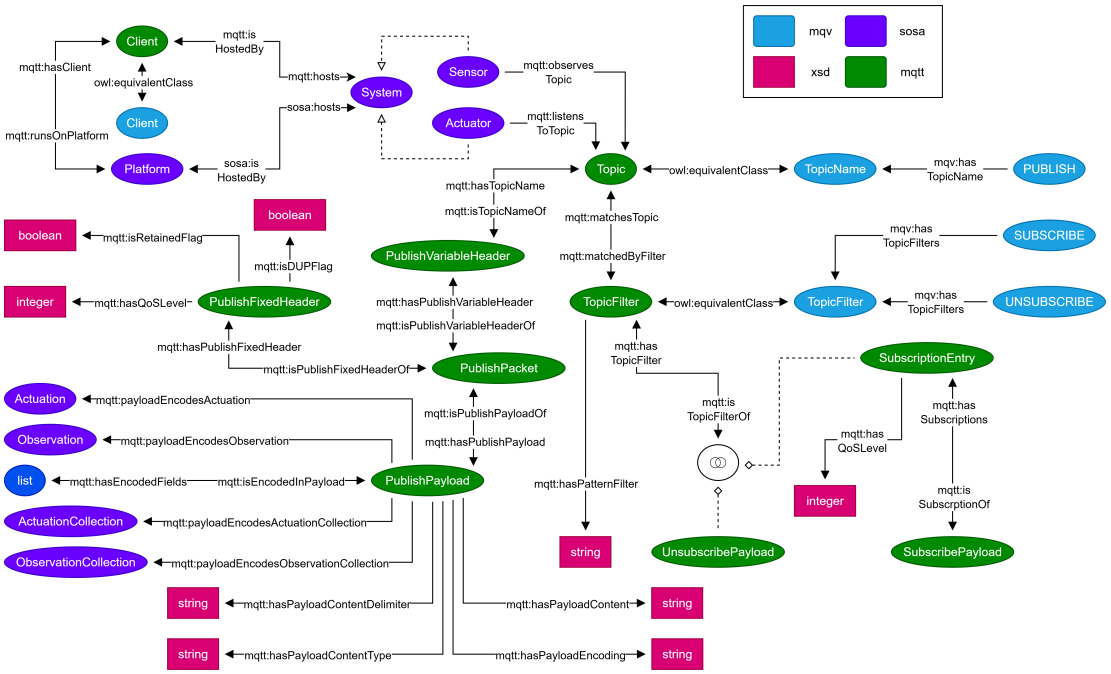

Figure 3: Topic and Payload relations

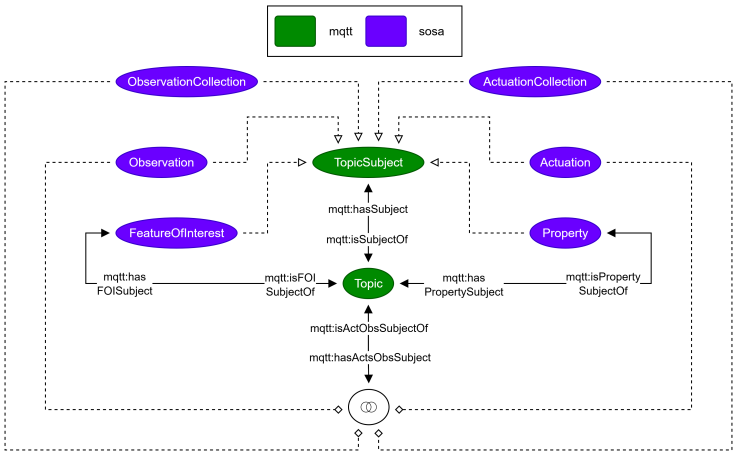

Figure 4: Topic

3.3. Design Patterns

The **MQTT4SSN** ontology integrates **MQV** vocabulary in the **SOSA/SSN** ontology stack to provide a semantic layer for modeling sensor observations and actuator commands over MQTT-based infrastructures. This alignment bridges the gap between the low-level communication protocol MQTT and high-level semantic descriptions of IoT systems via **SSN/SOSA**. We aligned our **MQTT4SSN** with the lightweight core ontology **SOSA**, while also supporting integration with the more expressive **SSN** due to the modular structure of **SSN/SOSA**. The **MQTT4SSN** and **SSN/SOSA** alignment is depicted in figures 3, 4, while neglecting detailed visualization of **SSN/SOSA** object properties due to space constraints.

To achieve this, key classes and properties from the **MQV** vocabulary were aligned with newly defined or extended classes or properties within our MQTT namespace. Several core classes were mapped directly to their **MQV** counterparts using `owl:equivalentClass`. For example, `mqtt:NetworkConnection ≡ mqv:NetworkConnection`, `mqtt:Broker ≡ mqv:Server`, and `mqtt:Client ≡ mqv:Client`. These equivalences ensure that MQTT network entities modeled in **MQTT4SSN** are semantically consistent with the existing **MQV** definitions, allowing direct inferencing across both ontologies and enabling future interoperability. The alignment was also applied to messaging constructs, such as `mqtt:Topic ≡ mqv:TopicName` and `mqtt:ControlPacket ≡ mqv:ControlPacket`.

Beyond class equivalences, the ontology also maps various object properties using `owl:equivalentProperty`. For instance, `mqtt:sendsControlPacket ≡ mqv:sendsControlPacket`, `mqtt:hasTopicName ≡ mqv:hasTopicName`, and `mqtt:hasPayload ≡ mqv:hasPayload` ensure that message composition and delivery semantics remain consistent regardless of whether they are interpreted from the perspective of **MQTT4SSN** or **MQV**. These alignments allow for symmetric entailment: for any triple using an aligned property in one ontology, a corresponding assertion in the other ontology is logically entailed.

Beyond class and object property alignments, the ontology also defines a range of datatype properties that describe literal characteristics of MQTT communication entities such as clients, brokers, payloads, and headers. Several of these properties are also explicitly aligned with their **MQV** counterparts using `owl:equivalentProperty`. For example, `mqtt:isRetainedFlag ≡ mqv:hasRetainFlag` and `mqtt:isDUPFlag ≡ mqv:hasDUPFlag` ensure that control flags in MQTT publish messages are semantically interchangeable between the two namespaces.

In contrast, other datatype properties deliberately avoid equivalence due to structural mismatches. The most notable example `mqtt:hasQosLevel`, which expresses the Quality of Service level as a constrained integer datatype (with values between 0 and 2). While **MQV** models this concept using the object property `mqv:hasQoSFlag`, the type mismatch between the datatype property and the object property prevents a clean alignment. As such, no equivalence is declared, preserving semantic consistency and avoiding reasoning errors caused by incompatible property types. We left a corresponding comment on the definition of the datatype property.

We add datatype properties to describe essential identifiers and configuration details of network components. For instance, `mqtt:hasClientID`, `mqtt:hasHostAddress`, `mqtt:hasPortNumber`, and `mqtt:usesTLS` provide literal values such as client identifiers, broker addresses, and encryption status, all of which are critical for modeling MQTT network behavior at the protocol level that MQV did not implement. Similarly, payload-specific properties like `mqtt:hasPayloadContent`, `mqtt:hasPayloadContentType`, `mqtt:hasPayloadContentDelimiter`, and `mqtt:hasPayloadEncoding` describe the structure and format of transmitted data.

The ontology integrates the **QUDT** ontology to represent measurement results in a precise and interoperable manner. Observation values and payload metadata are expressed as `qudt:QuantityValue` instances, which represent a `qudt:value` annotated with standard units from the **QUDT** vocabulary (e.g., `qudt:hasUnit unit:MilliM`, `qudt:hasUnit unit:Byte`). Using **QUDT**, we ensure that units are machine-readable, semantically precise, and avoid ambiguity in data interpretation. **QUDT** enables seamless integration with domain ontologies that rely on quantitative measurements and allows consistent handling of observational and technical metadata, like payload sizes.

4. Evaluation

The ontology evaluation is divided into verification 4.1 and validation 4.2. Verification takes the previously defined RQs and CQs as requirements and proves their presence in the ontology, represented by entities and their relations. For validation, the modeled ontology is instantiated with real-world individuals to determine whether it provides sufficient utility in the application. During development and finally, we checked the ontology with the OntOlogy Pitfall Scanner! (OOPS!) [22], locally installed

with the latest Docker Image[2]. The final OOPS! evaluation detects only minor pitfalls, which warn about unconnected ontology elements (P04), missing annotations (P08), and missing inverse relationships (P13). The warnings P04 and P08 refer to imported ontologies, whereas we explicitly modeled the identified relationships in P13 as non-inverse. The full OOPS! evaluation report[3] is available in the ontology documentation.

4.1. Ontology Verification

During the ontology engineering of **MQTT4SSN**, RQs and CQs were previously defined, representing the ontology's requirement profile. The RQs define the coverage of the general purpose in the application. The CQs determine specific entities and their relations for the model design. Firstly, we identified CQs to address essential components of the MQTT message protocol. Secondly, we determined CQs to address the sensing and communication semantics gap. We categorized the CQs into six subjects to structure them according to their common topics. The complete set of competency questions is available on GitHub[4]. Table 1 shows an excerpt of one CQ per subject with entities that represent the response.

Table 1
An excerpt of one CQ per topic and the corresponding entities they are answered by.

Subject	CQ	Question	Entities
Network and Participants	CQ1_03	Which MQTT participants (Clients/Brokers) use which MQTT NetworkConnections?	(Client - isInitiatedBy - NetworkConnection - isAcceptedBy - Broker)
Topics and Topic Filters	CQ2_01	Which SOSA sensors publish to which MQTT Topics (observations)?	(Sensor - observesTopic - Topic - has Subject - TopicSubject - Observation)
Control Packets	CQ3_01	Which MQTT Clients send which MQTT Control Packets?	(Client - sendsControlPacket - ControlPacket)
Header and Payloads	CQ4_05	Which MQTT Control Packets have which PacketIdentifier?	(ControlPacker - hasVariableHeader - VariableHeader - hasPacketIdentifier - unsignedShort)
Subscriptions	CQ5_01	Which MQTT SUBSCRIBE payloads contain which SubscriptionEntries?	(SubscribePayload - hasSubscriptions - SubscriptionEntry)
Payload content	CQ6_05	What is the size of a MQTT payload?	(Payload - hasPayloadSize - QuantityValue)

We can confirm that all our originally defined CQs are modeled in our ontology, which underlines our conceptual completeness.

4.2. Ontology Validation

To fully validate **MQTT4SSN** for real-world applications, it is essential to instantiate the ontology with individuals from a dataset, such as an industrial production process scenario using the MQTT message protocol. The next step is to formulate SPARQL queries to test the ontology against the dataset and execute them. The confirmation of expected answers validates the ontology with conceptual completeness.

[2]https://hub.docker.com/r/mpovedavillalon/oops, accessed: September 12, 2025
[3]https://doernern.github.io/MQTT4SSNOntology/documentation/OOPSevaluation/oopsEval.html
[4]https://github.com/doernern/MQTT4SSN/blob/main/CQs/MQTT4SSN_CQs.txt

We have neither acquired a dataset nor created individuals for validation yet. In the future, we plan to investigate this in a Raspberry Pi simulation setup to generate test data and validate **MQTT4SSN** with appropriate SPARQL queries.

5. Conclusion and Future Work

From the outset, **MQTT4SSN** was conceived as an extension of the well-established **SSN/SOSA** ontology, incorporating a semantic representation of the MQTT message protocol. While **SSN/SOSA** provides a robust framework for modeling sensing systems, including sensors, actuators, and their relationships to observations, properties, and features of interest, **MQTT4SSN** complements this foundation by explicitly modeling the structure and metadata of transmitted data. The ontology captures the essential elements of MQTT, such as the network entities broker and client, the various control packets and their payloads, the topics that organize communication, and the interrelations between these components. This integration bridges the gap between sensor semantics and M2M communication semantics, supporting end-to-end traceability and semantic interoperability across distributed IoT systems.

In addition, **MQTT4SSN** describes a payload content with a specific content type and character encoding, making payload descriptions meaningful and machine-interpretable. Transport-specific features such as QoS levels and the retain flag are also incorporated to enable traceability of message delivery. Despite the incompleteness of the **MQV** draft ontology, **MQTT4SSN** successfully adapts selected concepts and establishes partial semantic mappings to facilitate compatibility and reuse.

Several promising directions are still open for further development of **MQTT4SSN**. In our work, we focused exclusively on MQTT version 3.1.1. Future work could extend this scope to include the more recent MQTT v5 specification, which introduces enhanced message properties, session control, and user-defined metadata features. Furthermore, the MQTT-SN draft could extend the ontology, which adapts MQTT for non-TCP/IP networks such as UDP, Zigbee, or Bluetooth.

Another promising direction is the automated derivation of topic names. While **MQTT4SSN** currently supports modeling the relationships between topic naming and **SOSA** elements such as FeatureOfInterest, Property, Actuation, ActuationCollection, Observation, and ObservationCollection, an automated method for deriving meaningful topic names from these linked semantic entities could significantly reduce configuration effort and improve consistency. Similarly, the PayloadContent could be semantically derived from **SOSA** concepts. For instance, if the PayloadContentType is a specified application/json type, its structure could be automatically inferred from a combination of the aforementioned **SOSA** elements, enhancing semantic interoperability and reducing manual modeling.

We evaluate the ontology only to a limited extent. A valuable next step would be establishing a simulation environment using Raspberry Pis to emulate processes in industrial automation scenarios that communicate via MQTT to validate the proposed ontology comprehensively. This setup could be a testbed for validating the integration and mappings between live MQTT traffic and the **MQTT4SSN** ontology under realistic conditions.

Acknowledgments

This research, as part of the project EKI[1], is funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr, which we gratefully acknowledge. dtec.bw is funded by the European Union – NextGenerationEU.

Resource Availability Statement

The **MQTT4SSN** ontology[5] and the competency questions[4] are available on GitHub [23] and Zenodo (DOI: 10.5281/zenodo.16704302). All resources are licensed under Creative Commons Attribution-

[5] https://doernern.github.io/MQTT4SSNOntology/MQTT4SSN.owl

NonCommercial-ShareAlike 4.0 International. An enriched documentation[6] of the MQTT4SSN ontology was automatically generated with the help of the WIDOCO wizard [24].

Declaration on Generative AI

During the preparation of this work, the author used ChatGPT and Grammarly in order to: Grammar and spelling check, Paraphrase, and reword. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] W. Li, G. Tropea, A. Abid, A. Detti, F. Le Gall, Review of standard ontologies for the web of things, in: 2019 Global IoT Summit (GIoTS), 2019. doi:10.1109/giots.2019.8766377.

[2] G. Kim, S. Kang, J. Park, K. Chung, An MQTT-Based Context-Aware Autonomous System in oneM2M Architecture, IEEE Internet of Things Journal 6 (2019) 8519–8528. URL: https://ieeexplore.ieee.org/document/8726084/. doi:10.1109/jiot.2019.2919971, publisher: Institute of Electrical and Electronics Engineers (IEEE).

[3] M. Markovic, P. Edwards, Enhancing transparency of mqtt brokers for iot applications through provenance streams, in: Proceedings of the 6th International Workshop on Middleware and Applications for the Internet of Things, 2019, pp. 17–20. doi:10.1145/3366610.3368099.

[4] T. C. Piller, A. Khelil, Semsub: Semantic subscriptions for the mqtt protocol, in: 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), 2020. doi:10.1109/wf-iot48130.2020.9221477.

[5] T. C. Piller, D. M. Merz, A. Khelil, Mqtt-4est: Rule-based web editor for semantic-aware topic naming in mqtt, in: 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), 2022. doi:10.1109/ccnc49033.2022.9700533.

[6] F. Zhang, M. Aubart, R. Becker, B. Özcan, J. Blankenbach, Real-time sensor data integration for bim-based hydraulic structure monitoring, Unpublished Manuscript (2025).

[7] A. Banks, R. Gupta, Mqtt version 3.1.1, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html, 2014. OASIS Standard. Latest version: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

[8] A. Banks, E. Briggs, K. Borgendale, R. Gupta, Mqtt version 5.0, https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html, 2019. OASIS Standard. Latest version: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[9] OASIS MQTT Technical Committee, Mqtt-sn version 1.2, https://groups.oasis-open.org/higherlogic/ws/public/download/66091/MQTT-SN_spec_v1.2.pdf/latest, 2020. Working Draft. OASIS Standard. Latest version: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[10] K. Sahlmann, F. Mikolajczak, B. Schnor, Interoperability in the iot – an evaluation of the semantic-based approach, arXiv (2022). doi:10.48550/arXiv.2203.14585, arXiv:2203.14585.

[11] W3C Semantic Sensor Network Ontology, Semantic sensor network ontology (ssn), https://www.w3.org/TR/vocab-ssn/, 2017. W3C Recommendation.

[12] W3C Spatial Data on the Web Interest Group, Ssn/sosa ontology draft, https://w3c.github.io/sdw-sosa-ssn/ssn/, 2024. Draft version.

[13] W3C Web of Things, Mqtt-to-rdf ontology and binding templates, https://w3c.github.io/wot-binding-templates/bindings/protocols/mqtt/ontology.html, 2024.

[14] QUDT.org, Quantities, units, dimensions and data types (qudt), https://www.qudt.org/, 2024.

[15] M. Bodenbenner, Providing FAIR Sensor Data Models Using Semantic Web Technologies and Ontologies, Ph.D. thesis, University Thesis, 2025.

[16] ETSI SmartM2M, Saref - smart applications reference ontology, https://saref.etsi.org/extensions.html, 2024.

[6] https://doernern.github.io/MQTT4SSNOntology/documentation/index-en.html

[17] FIESTA-IoT Consortium, Fiesta-iot ontology, https://github.com/fiesta-iot/ontology, 2024.
[18] R. Agarwal, D. G. Fernandez, T. Elsaleh, A. Gyrard, J. Lanza, L. Sanchez, N. Georgantas, V. Issarny, 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. doi:10.1109/WF-IoT.2016.7845470.
[19] IRIT - MELODI Team, Iot-o: An ontology for semantic interoperability of iot resources, https://www.irit.fr/recherches/MELODI/ontologies/IoT-O.html, 2024.
[20] W3C Submission, Iot-lite ontology, https://www.w3.org/submissions/iot-lite/, 2024.
[21] IMT Atlantique, Semantic smart sensor network (s3n) ontology, https://recherche.imt-atlantique.fr/info/ontologies/sms/s3n/, 2024.
[22] M. Poveda-Villalón, A. Gómez-Pérez, M. C. Suárez-Figueroa, OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation, International Journal on Semantic Web and Information Systems (IJSWIS) 10 (2014) 7–34.
[23] N. Doerner, M. Maleshkova, doernern/MQTT4SSNOntology: MQTT4SSN v1.0.0: Pre-publication release, 2025. URL: https://github.com/doernern/MQTT4SSNOntology. doi:10.5281/zenodo.16704302.
[24] D. Garijo, Widoco: a wizard for documenting ontologies, in: International Semantic Web Conference, Springer, Cham, 2017, pp. 94–102. URL: http://dgarijo.com/papers/widoco-iswc2017.pdf. doi:10.1007/978-3-319-68204-4_9.