Monitoring, IoT Devices, and Semantics

Marc Vila*†, Maria-Ribera Sancho*‡, and Ernest Teniente*
*InLab FIB, Universitat Politècnica de Catalunya, Barcelona, Spain
{marc.vila.gomez,maria.ribera.sancho,ernest.teniente}@upc.edu
†Worldsensing, Barcelona, Spain
‡Barcelona Supercomputing Center, Barcelona, Spain

***Abstract*—Efforts to improve Internet of Things (IoT) device interoperability for monitoring are still required. This demo paper proposes monitoring infrastructure safety and security with the use of semantics. We make use of an ontology we proposed for interoperability in the IoT, the Connectivity Management Tool Semantics (CMTS) ontology. We demonstrate its use and the advantages it provides by monitoring a bridge, a crucial infrastructure that must be verified in near real-time. Two Raspberry Pi devices with sensors are used to monitor the inclination and vibration of the bridge, sending the data to a cloud server, which handles the ontology data-model. We also provide a web visualization tool, developed to assist users of the ontology to comprehend the status of the system's entities.**

***Index Terms*—IoT, Interoperability, Sensors, Semantics, Pervasive Computing**

I. Introduction

Various types of devices are increasingly used in the Internet of Things (IoT) to monitor entities throughout the world. What is apparent not only to the industry but also to academia is that a large number of these devices communicate data over the Internet, each with a different data format to provide the same semantic concept, being often very heterogeneous [1]. This results frequently in data incompatibilities, making it hard to extract the knowledge underlying the data [2]. Ongoing research in the *Interoperability of Things* is aimed at solving this problem by providing a homogenization of the communication characteristics or information among IoT devices and systems [3], [4], which is frequently accomplished through offering semantic interoperability [5].

Some years ago, both the writing and reading of the information to be communicated were done mainly by humans and it was indirectly solved by the use of a common spoken or written language. Nowadays, with the emergence of pervasive systems such as IoT, communication is mainly done Machine-to-Machine (M2M) [6]. Hence, the devices themselves are the ones obtaining the data, generating the information from this data, and sending it in real-time to other machine entities.

Figure 1 illustrates a communication between two devices that provide temperature measurements to a machine. However, there is no semantic interoperability between them, since one thermometer sends data in Celsius degrees and the other in Fahrenheit. There are two approaches to make them semantically compatible. The first one is to implement on the client machine a message translation mechanism that accepts both definitions and translates them locally. The second approach is to agree with both sensor manufacturers and communicate using a single measurement system, in this case, either Celsius or Fahrenheit. This second option is a step forward in terms of interoperability, since both entities share the same terminology and concepts.

The confluence of ontologies, semantics, and the IoT paradigm is being worked in [7]. As a result of this confluence, the monitoring of infrastructures via IoT devices has emerged. IoT devices are interconnected physical objects or entities equipped with sensors that allow them to observe, collect, and exchange data with other entities. They are used mainly as human consumables (such as wearables, health trackers, or home appliances) and also in industrial settings (e.g., Industry 4.0, Connected Cars, or Smart Cities) [8].

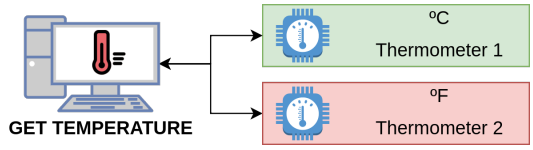


Fig. 1. Why do we need semantic interoperability?

In this demonstration, we focus on a previous work by Vila et al. [9], where a general ontology for monitoring entities using IoT devices was proposed. We exemplify it with Raspberry Pi devices that communicate measurements with our cloud server. These devices gather data from their wired accelerometer and temperature sensors, to have precise and real-time information about a bridge, as shown in Figure 2.

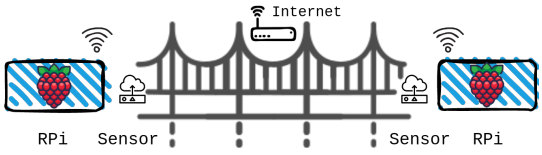


Fig. 2. Conceptual setup of the experiment

II. System Overview

A. Ontology

We use the Connectivity Management Tool Semantics ontology (CMTS) that we proposed in [9]. It is based on well-known ontologies, such as SSN/SOSA[1], GeoSPARQL[2] and OWL-Time[3]. Ontologies are used to represent the entities (i.e.

[1]https://www.w3.org/TR/vocab-ssn/
[2]https://www.ogc.org/standards/geosparql
[3]https://www.w3.org/TR/owl-time/

concepts) and relationships among them that exist within a domain. They are also helpful for monitoring the infrastructure of an IoT system because they allow for a clear and standardized representation of the different components and their interconnections. With a well-defined ontology, it becomes easier to analyze and understand the data generated by the system and to detect and diagnose problems that may arise.

Our ontology, summarized in Figure 3, contributes to the Interoperability of Things by providing semantics for the data management of IoT devices. It is built from two key concepts: the *Site*, i.e. a physical area being monitored, and the *Devices* installed to monitor. Furthermore, the ontology defines the *Sensor*s, a *Device* that may be either *Hardware* or *Software* and the *Gateways* and *Nodes*. It also enables the definition of the *Location* of the entities. Finally, measurements are named *Observations*, and these are the means to monitor the *ObservableProperty* of each *Sensor*.

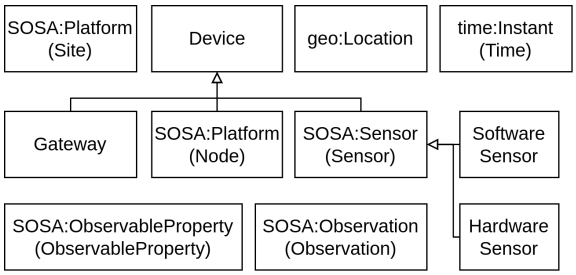


Fig. 3. Summary of components of the CMTS ontology [9]

B. Software Components

Our demo incorporates several software components which are encapsulated in functional modules since we make use of a microservices architecture, as seen in Figure 4. Each module has its own process that communicates with others using lightweight mechanisms, such as HTTP requests. With this, we empower the scalability, interoperability, modularity, and extensibility of the project.

When a user, either human or device, wants to communicate with our system, our *Kong*[4] API Gateway will be on the front line waiting to receive the request. When received, *Kong* handles it and responds in a good manner. That is, if the request aim is to serve an update in the frontend, *Kong* will contact with the *NGINX + React* service. If the request is to query or update the status of an element in our database, *Kong* will communicate with the our *Python FastAPI*[5] backend service. For instance, when a registered device sends a measurement to our Cloud and the backend server has to digest it. Our backend is made up of multiple endpoints, and it makes use of the CMTS ontology, implementing endpoints like the one that creates the *Sensors* or the one that allows measurement (*Observations*). Finally, once the data is received on our backend, it is ready to be used, manipulated, or displayed in our frontend.

[4]https://konghq.com/products/api-gateway-platform
[5]https://fastapi.tiangolo.com

For data storage, we use PostgreSQL, a relational database that can be optimized to handle data time series. Our frontend consists of React.js, a Javascript framework, where all the metadata of the system is observed. As a monitoring data visualization system, we use Grafana[6], which can be configured to be adapted to most of the IoT data types that we deal with. With all of this, we have a monitoring and visualization system with the data reported by the devices. We also provide a visualization tool in our Web frontend that shows entities instantiated in real-time in the system (Figure 5).

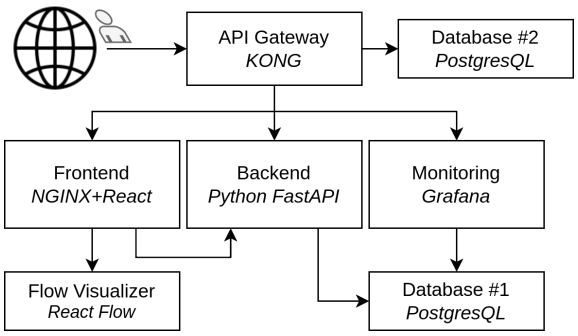


Fig. 4. Overview of the software components used

The system architecture is orchestrated and deployed using microservices by means of Docker and Docker-compose. In this way, we provide a systematic solution to automate a faster deployment of our components inside portable containers.

C. Hardware Components

We have a cloud server where all these components are running, and where humans and devices connect. Our cloud setup is a *GCP E2-small* instance with 1 vCore and 2 GB of RAM running Debian 11 Linux. Here, we deployed our own server code, developed according to the CMTS ontology.

We use as well two Raspberry Pi 4B devices with a GrovePi Shield[7] on top of each, with two *Grove - 6-Axis Accelerometer and Gyroscope*[8] sensors for monitoring. Raspberry Pi devices are capable of reaching the Internet by submitting measurements wirelessly to the *Cloud* server, as data is submitted using an HTTP API communication via their WiFi adapter.

III. Demonstration Scenario

We conducted our experiments using two Raspberry Pi located near a bridge, with the aim of monitoring its stability using accelerometer sensors. This setting is shown in Figure 6. The bridge is 3D printed with flexible material, that allows us to perform a variety of scenarios. On each side of the bridge, there is one sensor connected to the GrovePi shield on top of one Raspberry Pi. These sensors measure the inclination of the bridge's surface as well as the node temperature.

On the other hand, we have an instance of the system, shown in Figure 5, which is making use of the CMTS ontology.

[6]https://grafana.com
[7]https://www.seeedstudio.com/GrovePi.html
[8]https://www.seeedstudio.com/Grove-6-Axis-Accelerometer-Gyroscope.html

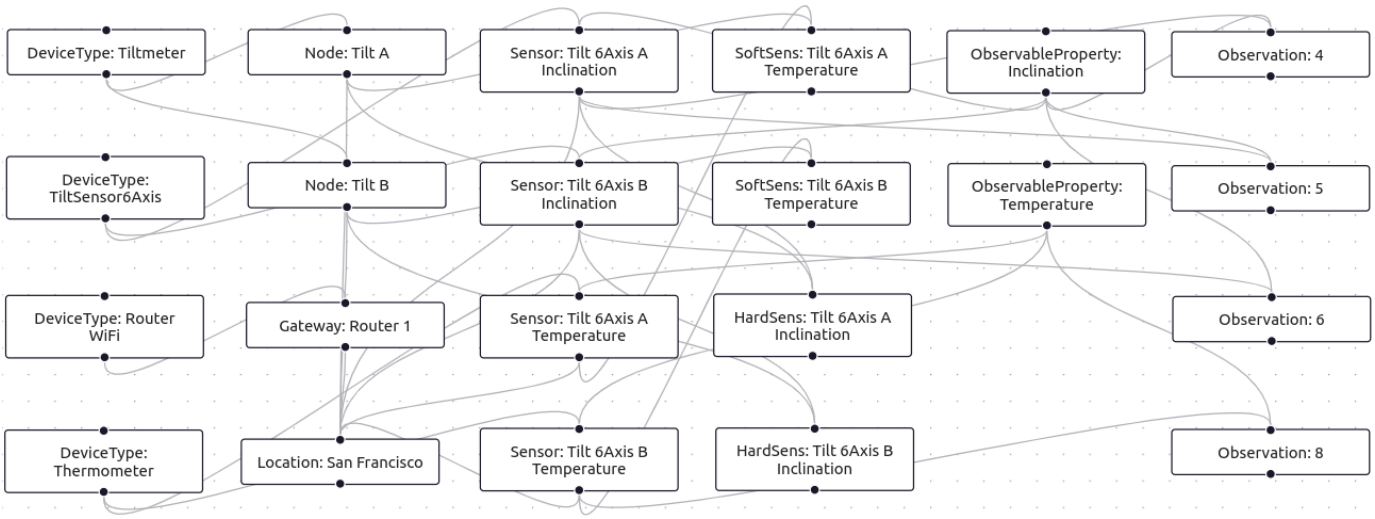


Fig. 5. Entities of the ontology and its relation to other entities

- Gateway: *Router 1* as the router that connects devices to the Internet.
- Node: *Tilt A* and *Tilt B* as the Raspberry Pi devices that hold the operative system and manage the measurements.
- HardwareSensor: *Tilt 6Axis A Inclination* and *Tilt 6Axis B Inclination* that are the *GrovePi* accelerometer and gyroscope sensors.
- SoftwareSensor: *Tilt 6Axis A Temperature* and *Tilt 6Axis B Temperature* that are the internal temperature processor sensors from the Raspberry Pi devices.
- ObservableProperty: *Inclination* as property to observe from the bridge. In addition to the *Temperature* observed.
- DeviceType: Four types of devices, *Router WiFi* for the *Gateways*, *TiltSensor6Axis* for the nodes and *Tiltmeter* and *Thermometer* for the Sensors.
- Location: All the mentioned entities are located in *San Francisco, CA, USA*.

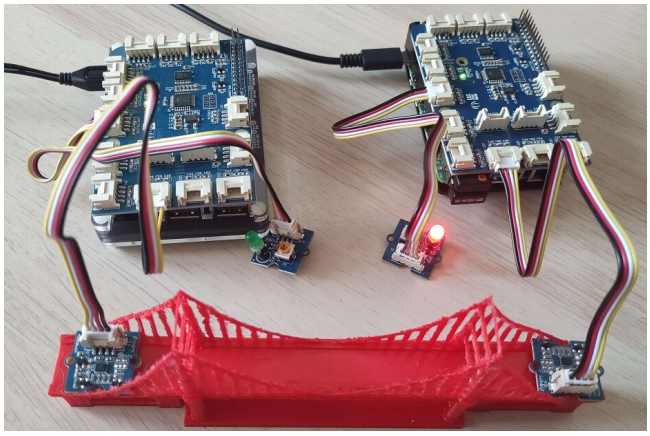
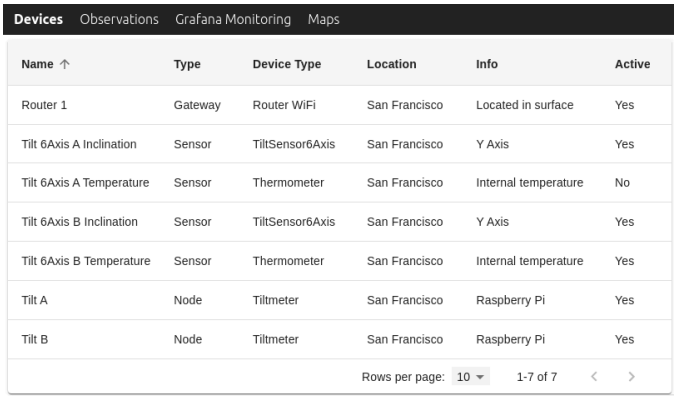


Fig. 6. Experimentation setup

The ontology, in addition to defining the format of the data to be sent, guides us in implementing the steps to take. So, the deployed system understands that the client —us in this demonstration— needs to perform the monitoring, by the definition of the elements through which it is accomplished.

The suggested frontend, in Figure 7, depicts the current state of the system. All of the entities that have been registered, as well as their metadata information, are shown.



Devices Observations Grafana Monitoring Maps

Name ↑	Type	Device Type	Location	Info	Active
Router 1	Gateway	Router WiFi	San Francisco	Located in surface	Yes
Tilt 6Axis A Inclination	Sensor	TiltSensor6Axis	San Francisco	Y Axis	Yes
Tilt 6Axis A Temperature	Sensor	Thermometer	San Francisco	Internal temperature	No
Tilt 6Axis B Inclination	Sensor	TiltSensor6Axis	San Francisco	Y Axis	Yes
Tilt 6Axis B Temperature	Sensor	Thermometer	San Francisco	Internal temperature	Yes
Tilt A	Node	Tiltmeter	San Francisco	Raspberry Pi	Yes
Tilt B	Node	Tiltmeter	San Francisco	Raspberry Pi	Yes

Rows per page: 10 1-7 of 7

Fig. 7. Our Devices tab frontend showing the onboarded entities

As stated in Section II-B, our system uses an API that is accessible via HTTP as its input mechanism. It is used to setup the complete frontend, as well as to enter the values of the measurements taken. The HTTP `GET`, `POST`, and `DELETE` methods are available to the user for each entity in the ontology. Listing 1 shows the code that should be provided in a `POST` request to our API's `/sensors` endpoint. This HTTP call is the one that generated the fourth row in Figure 7.

```
{
    "name": "Tilt 6Axis B Inclination",
    "device_type": "TiltSensor6Axis",
    "observable_property": "Inclination",
    "type": "hardwaresensor",
    "location": "San Francisco",
    "info": "Y Axis"
}
```

Listing 1. HTTP body to create a Sensor

In the *Observations* tab, there is a table with the observations and the sensors that measured them. When a sensor is selected, a monitoring view similar to the one illustrated in Figure 8 is shown. In this scenario, the graph of the *Tilt 6Axis B Inclination* sensor is shown between 15:51:00h and 15:54:00h, with the sensor reporting an acceleration on the x-axis between -1900 and -1800. This sensor measures between -32k and +32k, with 0 being neutral inclination. In our use case, the sensor is tilted to one side on the x-axis. Furthermore, around 15:52:45, the sensor detects vibration. This vibration lasts until 15:53:15 approximately.

The data of the experiment is reported by the components depicted in Figure 6, where the devices communicate information to our cloud server on a regular basis in the format prescribed by the CMTS ontology seen in Listing 1 and Listing 2 as HTTP communication examples with our system.

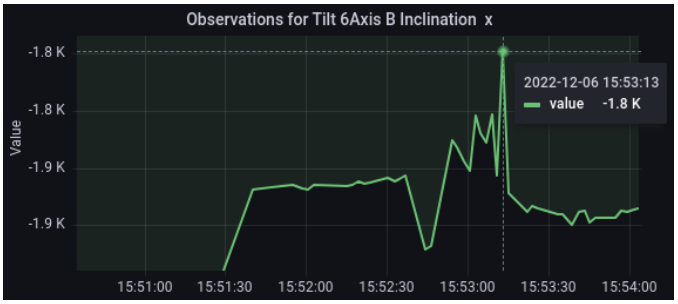


Fig. 8. Our Grafana frontend and the *Tilt 6Axis* for the X-Axis *Observations*

Listing 2 shows how data from the *Tilt 6Axis B Inclination* sensor was provided to the platform using a `POST` request to the `/observations` endpoint. Furthermore, the user can send the reading time. If the field is empty, the system will use the reception time as the reading time of the sensor. With that affirmation is how the Figure 8 depicts a time series based on a combination of several HTTP calls.

```
{
    "sensor_name": "Tilt 6Axis B Inclination",
    "observable_property": "Inclination",
    "value_int": -1950
}
```

Listing 2. HTTP body to submit an Observation

A working code for the entire experiment is available in https://github.com/worldsensing/demo-monitoring-using-iot-devices-and-semantics. This code orchestrates and manages the entire architecture as explained in Section II-B. It also contains the code that runs on the Raspberry Pi, to enable the reception of data from the physical sensors connected to it and transfers that data to the cloud backend. Additionally, the documentation schema is presented, see Figure 9, alongside the code when it is deployed in OpenAPI[9] format so that other developers may easily grasp the interface mechanisms, facilitating its further use.

[9]https://www.openapis.org

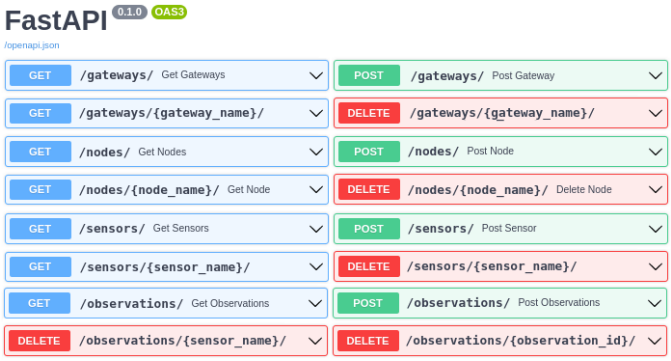


Fig. 9. Extract from the documentation generated using OpenAPI

In our demo at PerCom 2023, we will showcase the capabilities of our ontology by allowing users to monitor entities by displacing the sensors. We will demonstrate the working framework at the booth. Interested participants will be able to input measurements into the framework via requests made by Raspberry Pi devices and also to add their own elements to the system for custom monitoring.

IV. Conclusion

In this demo we contribute to the achievement of data homogeneity in the IoT domain by providing an implementation of the CMTS ontology in a fully functional framework, and by allowing users to manage and monitor their IoT devices with a minimal configuration. We also provide a practical experiment to monitor a bridge using two IoT devices which shows the advantages of developing applications according to CMTS.

Acknowledgments

This work is partially funded by Industrial Doctorates from Generalitat de Catalunya (2019 DI 001). SUDOQU project, PID2021-126436OB-C21 from MCIN/AEI, 10.13039/ 501100011033, FEDER, UE. Thanks to Xavier Vilajosana for his help in this work.

References

[1] P. Barnaghi et al. Semantics for the Internet of Things: Early Progress and Back to the Future. Int. J. Semant. Web Inf. Syst. 8(1), 1–21, 2012.
[2] M.A. Razzaque, M. Milojevic-Jevric et al. Middleware for Internet of Things: A Survey. IEEE Internet of Things Journal 3(1), 70-95, 2016.
[3] J. Kiljander et al., Semantic Interoperability Architecture for Pervasive Computing and Internet of Things. IEEE Access, vol. 2, 856-873, 2014.
[4] X. Su et al. Distribution of Semantic Reasoning on the Edge of Internet of Things. IEEE International Conference on Pervasive Computing and Communications (PerCom), 1-9, 2018.
[5] M. Noura, M. Atiquzzaman, and M. Gaedke. Interoperability in Internet of Things: Taxonomies and Open Challenges. Mobile Network Applications 24, 796–809, 2019.
[6] M. Elkhodr, S. Shahrestani, and H. Cheung. The Internet of Things: New Interoperability, Management and Security Challenges. International Journal of Network Security & Its Applications 8(2). 85-102, 2016.
[7] I. Szilagyi and P. Wira. Ontologies and Semantic Web for the Internet of Things - a survey. IEEE Industrial Electronics Society (IECON), 2016, pp. 6949-6954.
[8] P. Beckman, J. Dongarra et al. Harnessing the Computing Continuum for Programming Our World. Fog Computing: Theory and Practice, 215–230, 2020.
[9] M. Vila, MR. Sancho, E. Teniente, and X. Vilajosana. Semantics for Connectivity Management in IoT Sensing. Conceptual Modeling (ER). Lecture Notes in Computer Science 13011, 2021.